UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

DESARROLLO DE UN SISTEMA EMPOTRADO PARA EL CONTROL DE SISTEMAS ELECTRÓNICOS A TRAVÉS DE INTERNET

Mario Díaz Martínez
Tutor: Guillermo González de Rivera Peces

Febrero 2017

DESARROLLO DE UN SISTEMA EMPOTRADO PARA EL CONTROL DE SISTEMAS ELECTRÓNICOS A TRAVÉS DE INTERNET

AUTOR: Mario Díaz Martínez

TUTOR: Guillermo González de Rivera Peces

Trabajo realizado en el grupo

Hardware & Control Technology Laboratory

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Febrero de 2017

Resumen

En este Trabajo Fin de Grado se ha desarrollado un sistema electrónico basado en un sistema empotrado para ser manejado a través de internet. Este proyecto forma parte de la iniciativa de crear mostradores interactivos para ser inicialmente expuestos en la Escuela Politécnica Superior dentro de la Universidad Autónoma de Madrid.

Durante el proceso de desarrollo se estudiarán a fondo diferentes tecnologías actuales con el objetivo de construir la plataforma necesaria encargada de conectar al usuario con los equipos a mostrar.

Para ello se usará un sistema embebido que podrá ser controlado a través de internet. Una vez conectado a una página web, se mostrará el equipo y las funciones que puede realizar, permitiendo al usuario interaccionar en directo con las entradas y salidas de la plataforma.

El resultado de este proyecto será una plataforma totalmente accesible, que será la base para la conexión entre el usuario y el dispositivo a manejar. Se incluirá toda la documentación necesaria para poder acceder a los recursos.

Palabras clave

Sistema empotrado, Control remoto,Microcontrolador, Beaglebone Black, Servidor Web

Abstract

In this Final Project it has developed a system based on an electronic control system to be managed through the Internet. This project is part of the initiative to create interactive desks to be initially exhibited at the Escuela Politécnica Superior within the Universidad Autónoma de Madrid.

During the development process we studied a different technologies background with the aim of building the necessary platform.

To do this, use an embedded system that can be controlled through the Internet. Once connected to a web page, show the equipment and the functions they can perform, allowing the user to interact live with the inputs and outputs of the platform.

The result of this project will be a fully accessible platform, which will be the basis for the connection between the user and the device to be handled. All the documentation necessary to access the resources will be included.

Keywords

Embedded system, Remote control, Microcontroller,Beaglebone Black, Web Server.

Agradecimientos

En primer lugar me gustaría dar las gracias a Guillermo, mi tutor, por la confianza que ha depositado en mí para realizar este proyecto y por despertar mi interés y motivación en el tema de los microcontroladores y sistemas empotrados.

A mis compañeros de universidad, por hacerme creer en esta etapa de mi vida. A mis amigos de siempre por ser como son y acompañarme hasta el final. Gracias a todos por los buenos momentos, alegrías, cafés y por estar siempre ahí. Gracias Raúl, Bea Cervantes, Bea Cid, Guille, Marta, Riki, Lombar, Antonio, Sergio, Diego, Peli. Gracias por ser como sois.

A mi hermana Laura, por enseñarme a no rendirme, por ser siempre un apoyo y ser la mejor hermana. A mi familia, por todo vuestro apoyo ante los obstáculos en el camino, por la educación y valores que me habéis dado.

Mencionar especialmente a mí otra Laura, por confiar en mí, acompañarme hasta el final y darme el apoyo extra en los momentos más duros de esta increíble aventura.

Gracias por haberme enseñado tanto.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

El microcontrolador es uno de los logros más sobresalientes del siglo XX. Hace un cuarto de siglo tal afirmación habría parecido absurda. Pero cada año, el microcontrolador se acerca más al centro de nuestras vidas. Su presencia ha comenzado a cambiar la forma en que percibimos el mundo e incluso a nosotros mismos. Cada vez se hace más difícil pasar por alto el microcontrolador como otro simple producto en una larga línea de innovaciones tecnológicas.

La capacidad de los microprocesadores ha aumentado 10.000 veces en los últimos 25 años, y cada vez las prestaciones que ofrecen son mayores. Principalmente, el precio en el mercado puede rondar desde los diez euros hasta los cientos de euros dependiendo de las capacidades.

El atributomás grande del microcontrolador es que puede integrar inteligencia casi a cualquier artefacto. Se le puede entrenar para adaptarse a su entorno, responder a condiciones cambiantes y volverse más eficiente y que responda a las necesidades únicas de sus usuarios.

Aprovechando este avance tecnológico producido en los últimos años, así como la posibilidad de acceder a estas tecnologías a un bajo coste, y mi interés por los microcontroladores, se ha planteado la creación de unos mostradores interactivos en la EPS.

Por tanto, la motivación de este proyecto es el de desarrollar un sistema empotrado para el control de sistemas electrónicos a través de internet. Todo ello manejado por un microcontrolador.

1.2 Objetivos

El objetivo principal del proyecto es el diseño de una herramienta hardware y software, que permitirá el manejo y control de un equipo electromecánico de forma remota, a través de Internet, basado en un sistema empotrado.

Para ello, se usarán librerías de código libre que ayudaran a generar el código ejecutable, que podrán ser controlados de forma remota desde un servidor web montado en el sistema empotrado. Esto permitirá que alguien sin apenas conocimientos de programación, pero sí con leves nociones de diseño web, pueda construir una página web sencilla para controlar de forma remota sistemas electrónicos conectados al microcontrolador elegido. Una vez diseñada la página web, sólo se tendrán que vincular los ejecutables al diseño web implementado.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- En el primer capítulo se ha realizado una breve introducción del trabajo, así como la motivación y los objetivos.
- En el segundo capítulo se expone el estado del arte de este proyecto, estudiando los diferentes sistemas de control remoto, la estructura de nuestro microcontrolador, el uso de los sistemas embebidos y los actuales servidores web.
- En el tercer capítulo se explica el proceso de desarrollo de este proyecto, desde el estudio de las necesidades y componentes para el diseño de la plataforma, seguido de la elección del procedimiento de implementación software, hasta la elección de un sistema embebido para conseguir los objetivos fijados.
- En el cuarto capítulo se exponen las diferentes pruebas realizadas para comprobar el correcto funcionamiento del dispositivo.
- En el capítulo quinto se detallan las conclusiones y posibles líneas de trabajo futuro para este proyecto.
- Anexos y referencias.

2 Estado del arte

Antes de abordar el diseño y el desarrollo de este proyecto, es necesario estudiar previamente los productos y soluciones que existen actualmente en el mercado sobre plataformas de control remota para hacer una idea de cómo será nuestra plataforma.

2.1 Sistemas de control remoto

2.1.1 Domótica – X10

La domótica es el sistema capaz de automatizar una vivienda y gestionar los componentes que están integrados por medio de una comunicación cableada o inalámbrica. El control de los elementos conectados podrá ser de forma local o de forma remota.
Las aplicaciones puedes ir desde programación de eventos y ahorro energético, hasta sistemas de seguridad y confort entre otros. [1]

X10 es el protocolo de comunicaciones para control remoto de dispositivos eléctricos que nació en 1978 por Pico Electronics of Glenrothes (Escocia) y fue el primero de este tipo de tecnología.
Para su funcionamiento, hace uso de los enchufes eléctricos (220V – 110V), sin necesidad de cableado nuevo. La forma de comunicar con los diferentes elementos se basa en la transmisión de ráfagas de pulsos de RF (120 kHz) que representan información digital personalizada para cada elemento.

Puede funcionar correctamente para la mayoría de los usuarios domésticos. Es de código abierto y es el más difundido. Una desventaja es que es poco fiable frente a ruidos eléctricos.

Las plataformas personalizables que actualmente existen en el mercado, y que por tanto,nos permiten crear aplicaciones usando el protocolo de comunicaciones X10 para manejar de forma remotaelementos serían: Activehomepro, Misterhouse o Heyu entre otros.[2]

Figura 1 – Mando de control - X10

2.1.2 Hue – Phillips

Esta plataforma de control remota de luces, permite manejar todas las luces del hogar y crear eventos para simular que alguien esté en casa, despertarnos…

Este tipo de tecnología es posible gracias a Apple Homekit, esta plataforma nos permitirá hacer un control total de los elementos a través de unaaplicación o por el gestor de voz “Siri”.

Este sistema es “plug and play” de fácil instalación y uso, la desventaja de este producto es el precio, ya que es algo elevado para la mínima de cantidad de elementos que se pueden controlar. [3]

2.1.3 SorelMikroelektronik

Esta empresa alemana es probablemente la solución que más se acerca a nuestro principal objetivo. Sorel tiene a su disposición un conjunto de módulos de control “plug and play”, para que una persona sin apenas conocimientos pueda conectar a unmódulo, según un manual de instalación, cualquier elemento a controlar.
Estos módulos consisten en una pantalla LCD donde muestran la funcionalidad y unos pines externos donde conectar los elementos a controlar. Son bastantes intuitivos y de fácil instalación.
Control de calefacción, bombas de agua y bombas de calor entre otros son los principales productos de esta empresa. [4]

Figura 2 – Modulo control bomba de calor Sorel

3 Diseño

En el apartado de diseño explicaremos cómo se diseña esta plataforma de control remoto, describiendoel microcontrolador que usaremos, los modos de comunicación que usaremos para comunicar los sistemas electrónicos y de qué forma lo haremos.

3.1 Sistema embebido

Este proyecto debe ser controlado mediante un microcontrolador. Por ello en este trabajo, se ha optado por el uso de un sistema empotrado, ya que su amplia funcionalidad permite cubrir un conjunto de necesidades específicas para el diseño.
Estos dispositivos iniciaron su andadura como pequeños proyectos educativos como Raspberry o Arduino, y han evolucionado de forma que se han desarrollado como pequeños ordenadores; muchos de ellos incluyendo HDMI,audio o algunas características que podrían parecer impensables, como aceleradores de gráficos o procesadores independientes a tiempo real. Dado que estos sistemas tienen una gran capacidad de expansión porque incorporan multitud de entradas y salidas, el objetivo es desarrollar la plataforma que sea capaz de comunicarse y ser controlado mediante el sistema embebido.[5] [6] [26]

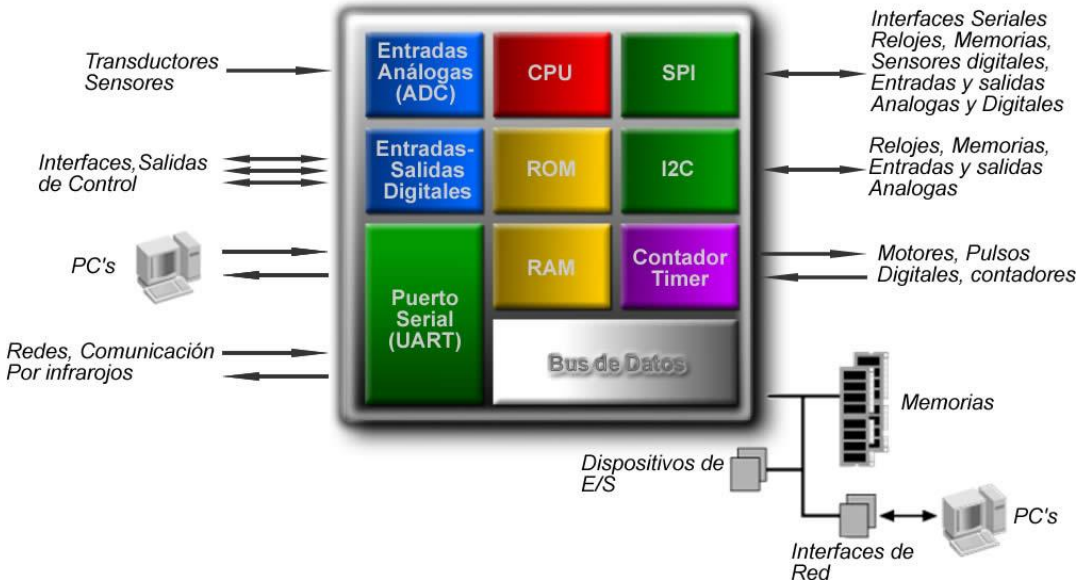


Figura 3 – Esquema de un sistema empotrado

Los principales fabricantes de este tipo de sistemas son Raspberry, Gumstix, PandaBoard y Beagleboard entre otros.

En concreto, este proyecto se va a desarrollar con BeagleBone Black, pero se va a presentar las principales características y prestaciones comparándola con otra placa competidora del mercado:

Raspberry Pi [7] [9]:

- Ventajas: Excelente tarjeta, potente, ideal para proyectos que requieren conectarse a una red. Existe una gran comunidad que aporta códigos, librerías. Se adecua perfectamente para proyectos basados en I2C, SPI o UART.

- Desventajas: No posee entradas analógicas ni PWM, lo cual implica el uso de más hardware para poder manejar este tipo de señales. Algunos componentes no son tan fáciles de conseguir y tal vez se deban adquirir, agregando coste al proyecto. Los voltajes que opera son de 3,3 V y no soporta voltajes de 5 V.

BeagleBone Black [7] [10]:

- Ventajas: Muy flexible a la hora de conectar sensores y actuadores directamente. Posee un potente procesador y un entorno Linux. Ideal para aplicaciones que requieran leer sensores y envío/despliegue de datos a internet.
- Desventajas: No posee todavía una comunidad tan amplia como Raspberry, pero poco a poco va creciendo. Las entradas analógicas que posee solo soportan 1,8 V, más voltaje puede dañar los pines de la tarjeta, por lo que para manejar más voltaje se debe emplear un circuito integrado extra.

Una vez hecha la comparativa y a sabiendas de las necesidades de nuestro proyecto vamos a optar por BeagleBone Black, principalmente porque tiene mucha más capacidad para interactuar con sensores (65 pines GPIO y 7 entradas analógicas), además de poseer un procesador más potente y más capacidad de almacenamiento.

3.1.1 Beaglebone Black

El sistema hardware elegido ha sido la placa BeagleBone Black, la misma que se muestra en la siguiente figura.

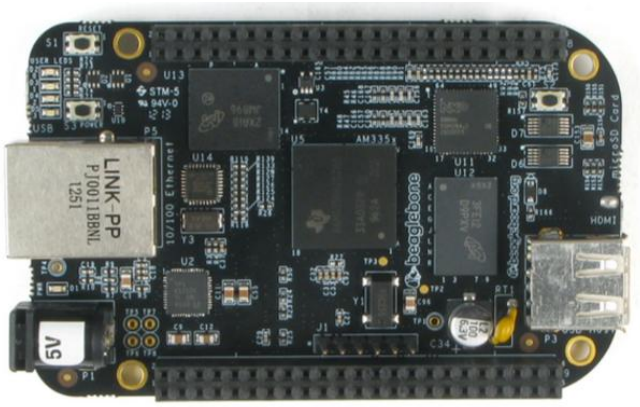


Figura 4 – Microcontrolador Beaglebone Black

A continuación, se analizan las características principales de esta placa que han permitido conocer los pasos necesarios para desarrollar el proyecto. El procesador, como se indica es el AM3558 Cortex A8 de 32 bits. Fabricado por Texas Instruments. La velocidad de éste, viene determinada por la alimentación entregada al dispositivo, por tanto es de 500MHz alimentada únicamente por vía USB y de 720 MHz con una fuente en DC. En la figura 3 se puede ver el sistema funcional del procesador. [5]

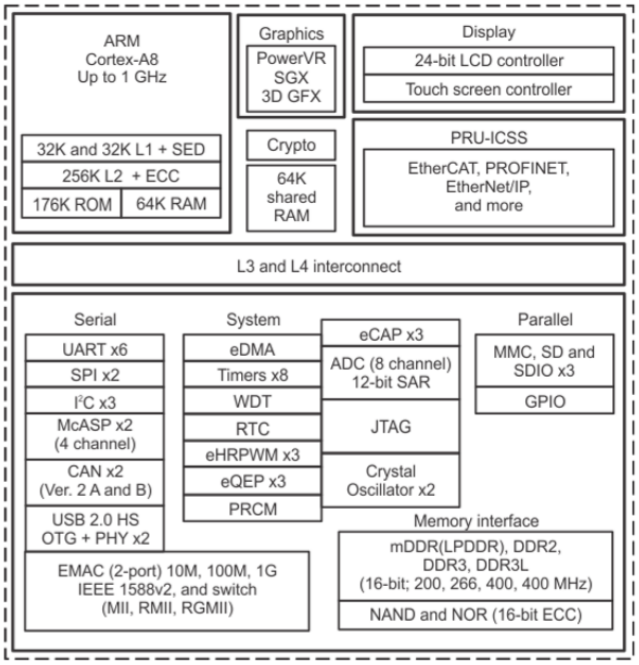


Figura 5 – Características AM3558 Cortex A8 de 32 bits

3.2 Modos de funcionamiento

La CPU deesta placa (AM335x 1 GHz ARM Cortex-A8)fué diseñada para quepudieran tener múltiples modos de funcionamiento tales como GPIO, PWM, ADC, I2C, SPI, PRU, HDMI, puertos USB y muchos otros. Los arquitectos de la CPU fueron conscientes de ello sabiendo que podrían ser usados por diferentes dispositivos y por diferentes diseños, consiguiendo así un microcontrolador personalizable.
A este tipo de dispositivos se les denominan On Chip Peripherals (de ahí el acrónimo OCP) y aunque son llamados periféricos en realidad están integrados en la propia CPU.

Cada dispositivo de OCP probablemente requerirá una o más entradas y salidas físicas en el exterior de la CPU de modo que el resto del sistema puede interactuar con él. El problema con esto es que, dados todos los dispositivos de OCP que se implementan, hay muchos más dispositivos OCP de Entada / Salida que pads de conexión en el exterior de la CPU.

Los diseñadores de la CPU llegaron a la conclusión de que la mayoría de las personas que implementaran la CPU nunca tendrán que utilizar todos los dispositivos de OCP a la vez, y por lo tanto llegaron a la siguiente solución.

Se estableció todo, de modo que los dispositivos de OCP compartiesen las almohadillas de Entrada / Salida de la CPU.

De esta manera, existen hasta 8 posibles usos de cualquier plataforma de CPU y el uso real es configurable en tiempo de ejecución. El nombre de los posibles usos se llama "modo”.

Llamaremos "pin" al dispositivo de OCP de Entrada / Salida de la línea interna de la CPU que se está haciendo referencia. Existen dos pads físicos (P8 y P9), llamados “header block”, de 46 pines cada uno conectados internamente a la CPU.

La CPU internamente hace la conmutación a los pines de OCP con el pad físico, a esto lo llamaremos PinMux.La configuración de PinMux (y por lo tanto "modo pinmux") para todos los dispositivos de OCP se establece por el devicetree en el arranque. Este modo puede ser cambiado por otro en el tiempo de ejecución y conmutar otro modo diferente.

El concepto de PinMux se puede asimilar como el de un simple switch rotatorio.

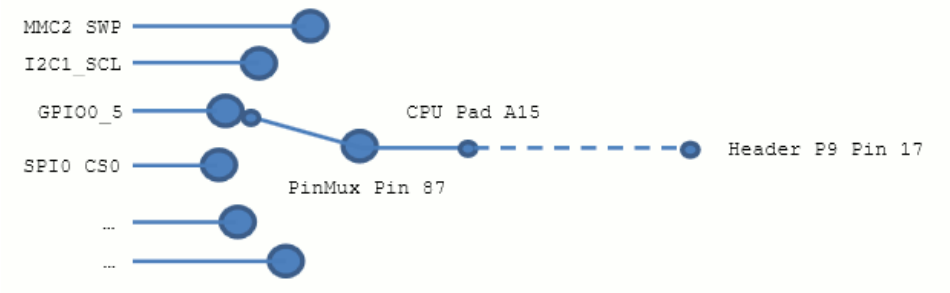


Figura 6 – Modos de funcionamiento - Pinmux

Debe tenerse en cuenta que no todas las patillas de la CPU van a los pads P8 y P9, algunos van directamente a los puertos USB, puerto HDMI o la memoria EMMC etc.

También destacar, que si dos dispositivos OCP utilizan la misma plataforma de la CPU, no se permitirá el uso de las dos al mismo tiempo. La razón es que ambos dispositivos de OCP comparten el mismo PinMux en diferentes modos. [5] [6]

Los modos de funcionamiento de las cabeceras de pines P8 y P9 se adjuntan en el Anexo de este proyecto. [27]

Aunque cada pin puede llegar a tener hasta 8 diferentes modos de funcionamiento. En este proyecto solo se van a definir cuatro tipos de modos, ya que suelen ser los más usados.

Los modos de funcionamiento que usaremos en este proyecto son:

- GPIO
- PWM
- ADC
- I2C

3.2.1 GPIO

GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General) es un pin genérico de la CPU del microcontrolador, cuyo comportamiento (incluyendo si es un pin de entrada o salida) se puede controlar (programar) por el usuario en tiempo de ejecución[11].

GPIO se utilizan en:

- Chips con pocos pines: IC, SoC, integrados y hardware a la medida, dispositivos lógicos programables (por ejemplo, FPGAs)

- Chips multifunción: gestores de energía, códec de audio, tarjetas de video.
- Aplicaciones embebidas como es el caso de este proyecto, donde se hace un uso intensivo de GPIO como por ejemplo, para la lectura de varios sensores ambientales (IR, de vídeo, la temperatura, la orientación de 3 ejes, aceleración), y para enviar la salida a motores de corriente continua (mediante PWM), audio o LCD o entre otros.

En el caso de este microcontrolador, existen hasta 65 pines con este modo de funcionamiento.

Para controlar este modo de funcionamiento será necesario configurar qué pin vamos a utilizar, si va a ser pin de entrada o salida y si se escribirá un "0" o un "1" lógico en ese pin.

3.2.2 PWM

La modulación por ancho de pulsos (PWM, pulse-width modulation) de una señal es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.[12]

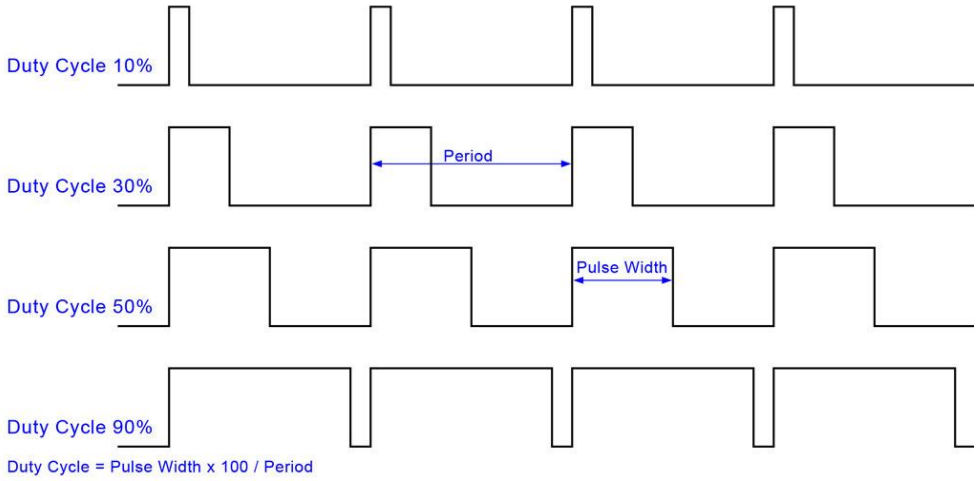


Figura 7 – Señal PWM

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$D = \tau/T$ D es el ciclo de trabajo

τ es el tiempo en que la señal es positiva (ancho del pulso)

T es el periodo de la señal

PWM se utiliza principalmente en el control de motores o servomotores.

En el caso de este microcontrolador, existen hasta 8 pines con este modo de funcionamiento.

Para controlar este modo de funcionamiento será necesario configurar qué pin vamos a utilizar, el periodo de la señal y el ciclo de trabajo.

3.2.3 ADC

El microcontrolador BeagleBone Black tiene la capacidad de hacer conversiones analógico-digitales. Esta capacidad es muy importante ya que actualmente existe gran capacidad de componentes analógicos.

ADC se utiliza principalmente es conversionesque nos resultaran útiles para la lectura de las señales de potenciómetros, sensores, drivers… que podrán ser posteriormente tratadas. [13]

En el caso de este microcontrolador, existen hasta 7 pines con este modo de funcionamiento.

Para controlar este modo de funcionamiento será necesario configurar qué pin vamos a utilizar, una lectura y su posterior conversión de la corriente que estamos leyendo en el pin.

Destacar que para este tipo de modo de funcionamiento, existe una alimentación diferente a la de la placa de 1.8 voltios.

3.2.4 I2C

I²C (Inter-Integrated Circuit) es un bus de datos serial desarrollado en 1982 por Philips Semiconductors.
Se utiliza principal e internamente para la comunicación entre diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos. [14]

El primer bus I²C se utiliza para la lectura de EEPROM y no puede ser utilizado para otras operaciones de E / S digitales sin interferir con esa función, se puede utilizar para añadir otros dispositivos I²C en las direcciones disponibles.

El segundo bus I²C está disponible y se puede configurar.

Esta placa es capaz de manejar hasta 2 buses de I²C con sus diferentes señales:

- SCL es la señal de reloj generada por el maestro.
- SDA es la señal de datos.

I²C se utiliza principalmente para que el microcontrolador o “maestro”, maneje a los diferentes dispositivos o “esclavos” conectados a él.

Para controlar este modo de funcionamiento será necesario configurar la dirección del dispositivo a conectar, la dirección del registro y el valor, tanto si se van a escribir como a leer datos.

3.3 Diseño web

El diseño web de la plataforma, basándose en una comunicación cliente - servidor, estará diseñada de la siguiente manera.

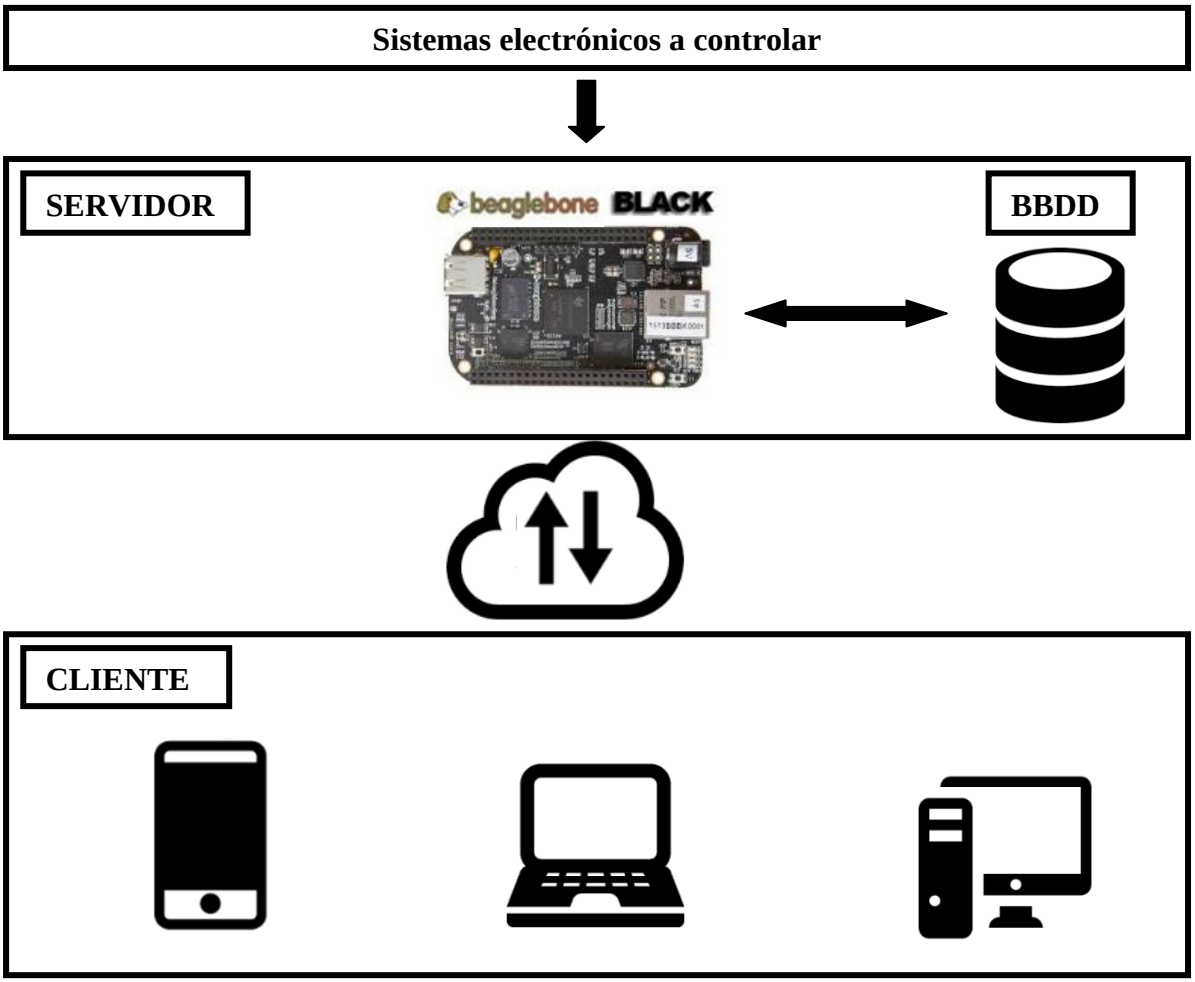


Figura 8 – Esquema diseño web

El usuario o cliente se conectará a una página principal (inicio.php) donde podrá introducir una serie de valores.
El servidor, en este caso el microcontrolador Beaglebone Black, recogerá estos valores y los ejecutará consiguiendo manejar de manera remota los dispositivos conectados a él.

3.3.1 Servidor Web

La plataforma para el control del microcontrolador será montada sobre un servidor web LAMP.

El servidor web LAMP es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas[15] [23]:

- Linux, el sistema operativo que está instalado en el microcontrolador.
- Apache, el servidor web.
- MySQL, el gestor de bases de datos.
- PHP, Python y C++ los lenguajes de programación usados.

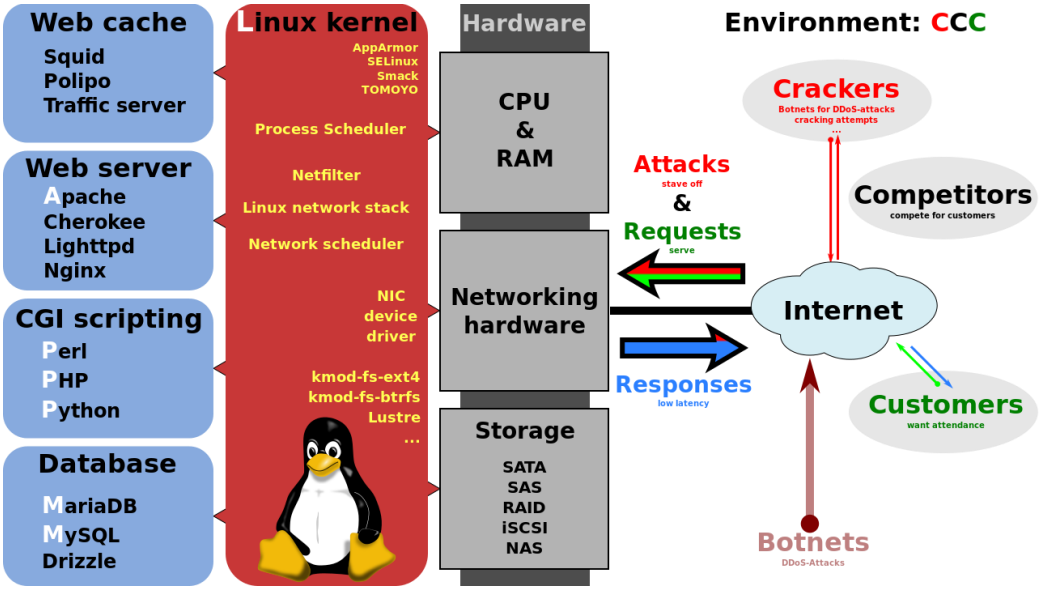


Figura 9 – Servidor web LAMP

La combinación de estas tecnologías nos permitirá construir la plataforma base para controlar el microcontrolador Beaglebone Black.

3.3.1.1 Distribución Linux

El sistema operativo usado en el servidor web LAMP es una distribución de Linux.
En nuestro caso, la imagen instalada es *Debian Image 2014-05*-14 que corresponde con un Debian GNU/Linux 7.11 (Wheezy) y puede ser descargada de la página oficial de beagleboard. [16][28] [29]

3.3.1.2 HTTP Apache

Para que el código de control de la BBB pueda ser controlado desde un navegador, se necesita un servidor web capaz de manejarlo [17].

Existe una gran cantidad de este tipo de servidores tales como: Lighttpd, Cherokee…Pero se ha elegido usar Apache, que es software de código abierto, por su popularidad, sencillez y gran comunidad online acerca de su uso y funcionamiento.

3.3.1.3 MySQL

Para que podamos leer o escribir datos externos, necesitamos de una base de datos capaz de mover la información. Esto es posible gracias al motor de base de datos MySQL. Este es un sistema de gestión de bases de datos de gran alcance usado para organizar y recuperar los datos. Un ejemplo posible es el de acumular datos de un sensor de temperatura y sacar una estadística de esos datos posteriormente[18].

3.3.1.4 PHP

Para que el código se pueda interpretar en el navegador, será necesario usar el lenguaje de programación adecuado. Esto será posible mediante la instalación y el uso de PHP.

PHP es un lenguaje de programación de código abierto que se utiliza para construir páginas web dinámicas. Es quizás el lenguaje más popular y el más utilizado a nivel mundial para el desarrollo web. [19]
Es un lenguaje que está muy bien documentado y se pueden encontrar un sinfín de ejemplos y tutoriales lo cual lo hace una muy buena opción para aprender y conocer sobre la programación.

Ventajas:

- Sintaxis similar a otros lenguajes.
- Lenguaje muy popular con una comunidad muy amplia.
- Rápido.
- Multiplataforma.
- Libre y gratuito.
- No requiere definición de variables.
- Tiene muchos frameworks que facilitan el desarrollo en este lenguaje.

Inconvenientes:

- Necesita un servidor para funcionar.
- Todo el trabajo se realiza en el servidor y mucha información o solicitudes pueden ser ineficientes.

3.3.2 Lenguajes de programación

3.3.2.1 Python

Python [20] es un lenguaje de programación que permite trabajar más rápido e integrar tus sistemas de una manera más eficiente. Se trata de uno de los lenguajes más populares y carismáticos y cuenta con una comunidad muy activa.
Es una opción más para realizar páginas web dinámicas, tiene el gran beneficio de ser un lenguaje multiplataforma por lo que tener problemas al utilizar una u otra plataforma es mínimo.
Ventajas:

- Libre y código fuente abierto.
- Lenguaje de propósito general.
- Multiplataforma.
- Orientado a objetos.

Inconvenientes:

- Los lenguajes interpretados suelen ser relativamente lentos.

3.3.2.2 C++

La intención de la creación del lenguaje C++ [21] fue el extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.
Posteriormente se añadieron facilidades de programación genérica, que se sumaron a los paradigmas de programación estructurada y programación orientada a objetos. Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

Una particularidad del C++ es la posibilidad de redefinir los operadores, y de poder crear nuevos tipos que se comporten como tipos fundamentales.
Ventajas:

- Ideal para sistemas robustos.
- Es multiplataforma.

Inconvenientes:

- No es popular para la creación de aplicaciones.
- Sintaxis compleja.

4 Desarrollo

En este capítulo se explica el código implementado en el desarrollo web visto por el usuario y el código de ejecución de cada modo de funcionamiento para la comunicación con los sistemas electrónicos a controlar.

4.1 Desarrollo web

4.1.1 Inicio.php

Este fichero contiene la página principal desarrollada en HTML con la que interactuará el cliente. En él aparece una serie de botones y cuadros de texto donde el usuario podrá introducir unos valores determinados. [22]

Estos valores, que forman parte de los argumentos de una serie de ejecutables, serán interpretados por el método POST en el servidor, ejecutando la acción correspondiente.

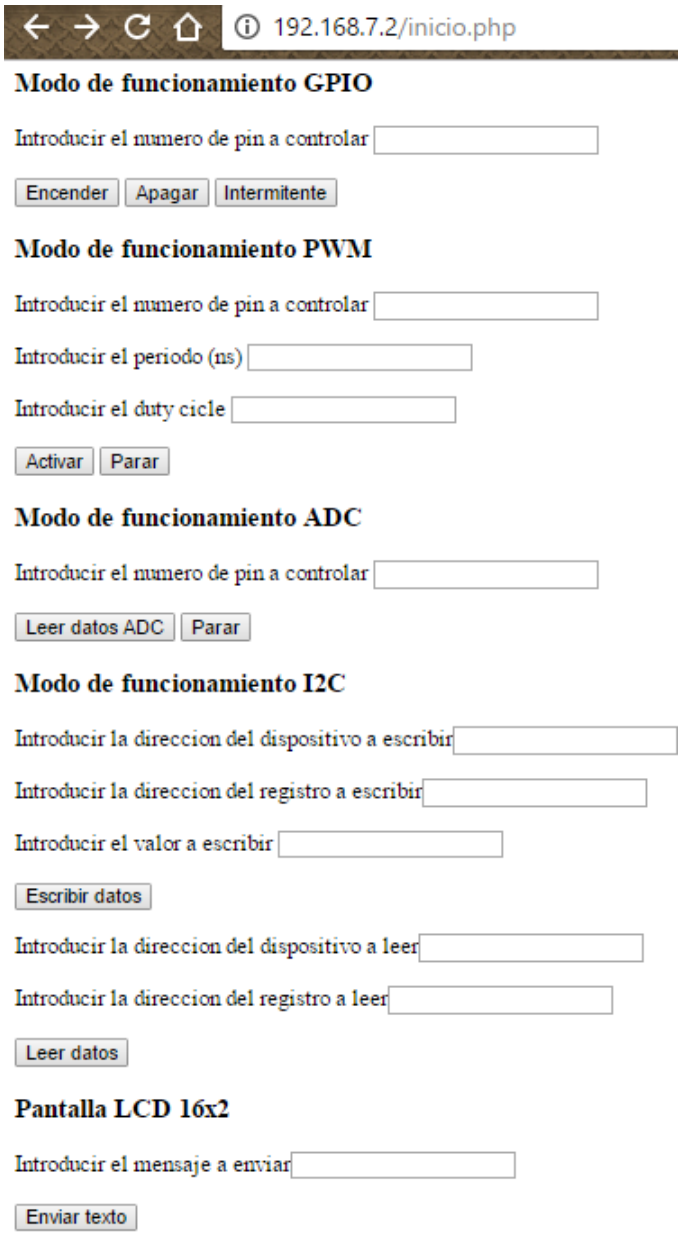


Figura 10 – inicio.php

4.2 GPIO

En el desarrollo de este modo de funcionamiento conectamos un LED a uno de los pines del microcontrolador Beaglebone Black según la siguiente figura.

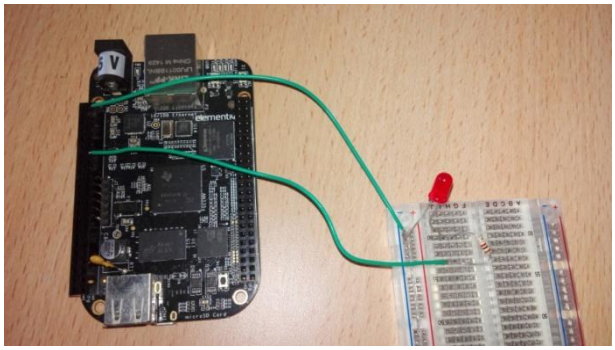


Figura 11- Desarrollo modo de funcionamiento GPIO

Para controlar este modo de funcionamiento utilizamos la librería de código y distribución libre BlackLib, la podemos encontrar en la página oficial de beagleboard.[24]

Para saber que pines pueden ser utilizados en este modo GPIO, utilizaremos la siguiente tabla.

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BUT	9	10	SYS_RESETn	GPIO_69	9	10	GPIO_68
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	GPIO_50	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA	GPIO_22	19	20	GPIO_63
GPIO_3	21	22	GPIO_2	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	GPIO_113	GPIO_86	27	28	GPIO_88
GPIO_111	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
GPIO_110	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	GPIO_81
AIN6	35	36	AIN5	GPIO_8	35	36	GPIO_80
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	GPIO_7	GPIO_74	41	42	GPIO_75
DGND	43	44	DGND	GPIO_72	43	44	GPIO_73
DGND	45	46	DGND	GPIO_70	45	46	GPIO_71

Tabla 1 - Pines de control GPIO

4.2.1 encenderGPIO.cpp

La compilación de este archivo con la librería anteriormente mencionada genera el ejecutable encenderGPIO.
Este ejecutable recibe por argumento el pin de control GPIO, modifica el pin como salida= “output” y establece el valor como high=”1”.

Se muestra a continuación una parte del código de encenderGPIO.cpp:

```
#include<iostream>
#include"BlackGPIO.h"
#include <string>
#include <fstream>
```

```
#include <unistd.h>
using namespace std;
using namespace BlackLib;
using std::string;

int main (intargc, char* argv[]) {

std::string myString = argv[1];
std::istringstream converter(myString);
int number = 0;
converter >> number;
intpinName = number;

BlackGPIOencenderGPIO(gpioName(pinName), output);
encenderGPIO.setValue(BlackLib::high);

return 0;
}
```

4.2.2 apagarGPIO.cpp

La compilación de este archivo con la librería anteriormente mencionada genera el ejecutable apagarGPIO.
Este ejecutable recibe por argumento el pin de control GPIO, modifica el pin como salida= "output" y establece el valor como low="0".

4.2.3 intermitenteGPIO.cpp

La compilación de este archivo con la librería anteriormente mencionada genera el ejecutable intermitenteGPIO.
Este ejecutable recibe por argumento el pin de control GPIO, modifica el pin como salida= "output" y establece el valor como high="1" y low="0" intermitentemente mediante un bucle for.

4.3 PWM

En el desarrollo de este modo de funcionamiento conectamos un LED a uno de los pines del microcontrolador Beaglebone Black según la figura 11.
Lo conectamos de esta manera ya que este pin comparte ambos modos de funcionamiento.

De esta manera podremos controlar el LED de diferente luminosidad cambiando el ciclo de trabajo de la señal PWM.

Para controlar este modo de funcionamiento utilizamos la librería de código y distribución libre BlackLib, la podemos encontrar en la página oficial de beagleboard. [24]

Para saber que pines pueden ser utilizados en este modo PWM, utilizaremos la siguiente tabla.

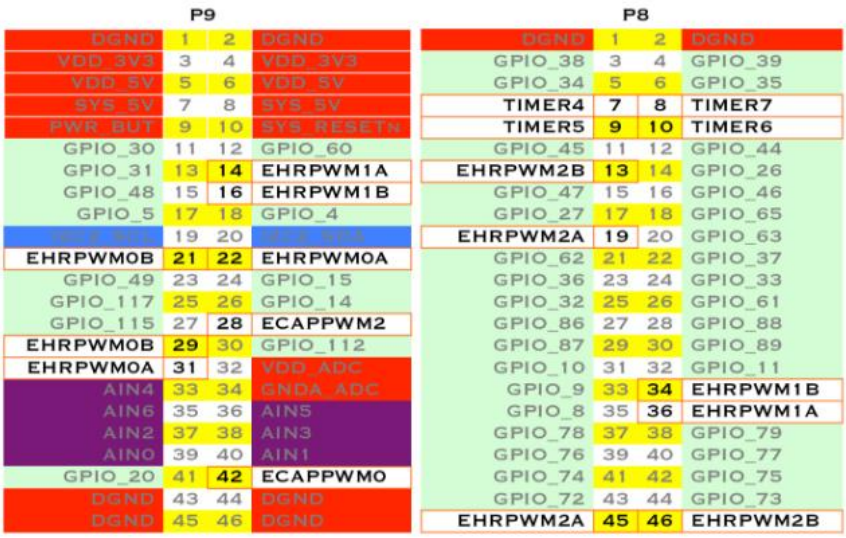


Tabla 2 – Pines de control PWM

Los argumentos de los ejecutables se corresponden con los siguientes pines de control PWM:

Pines de control PWM	Argumento del ejecutable
P8_13	0
P8_19	1
P9_14	2
P9_16	3
P9_21	4
P9_22	5
P9_42	6

Tabla 3 – Argumentos del ejecutable PWM

4.3.1 activaPWM.cpp

La compilación de este archivo con la librería anteriormente mencionada genera el ejecutable activaPWM.
Este ejecutable recibe por argumento el pin de control PWM, el periodo y el ciclo de trabajo. Tras esto, genera una señal según los argumentos 2 y 3 en el pin del argumento 1.

Las funciones usadas de la librería Blacklib son setDutyPercent() y setPeriodTime()

4.3.2 pararPWM.cpp

La compilación de este archivo con la librería anteriormente mencionada genera el ejecutable pararPWM.
Este ejecutable recibe por argumento el pin de control PWM y desactiva el modo de funcionamiento dejándolo en el modo inicial.

4.4 ADC

En el desarrollo de este modo de funcionamiento conectamos un potenciómetro a uno de los pines del microcontrolador Beaglebone Black según la siguiente figura.

De esta manera se puede ir leyendo el valor en cada instante.

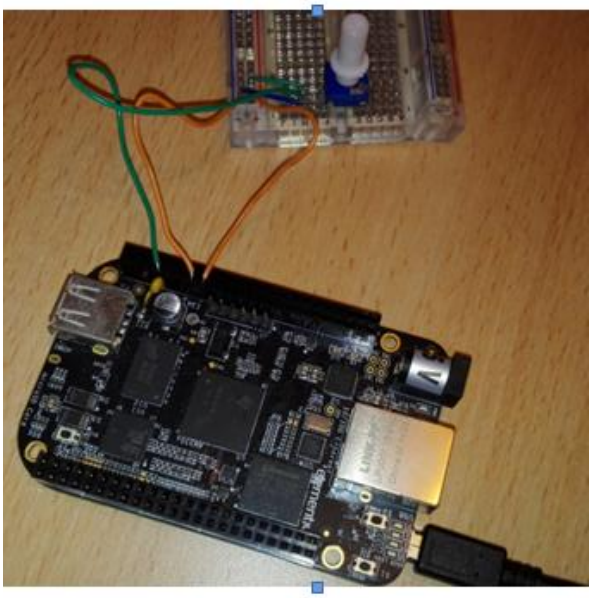


Figura 12 – Desarrollo modo funcionamiento ADC

Para controlar este modo de funcionamiento utilizamos la librería de código y distribución libre BlackLib, la podemos encontrar en la página oficial de beagleboard. [24]

Para saber que pines pueden ser utilizados en este modo ADC, utilizaremos la siguiente tabla.

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BUT	9	10	SYS_RESETn	GPIO_69	9	10	GPIO_68
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	GPIO_50	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA	GPIO_22	19	20	GPIO_63
GPIO_3	21	22	GPIO_2	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	GPIO_113	GPIO_86	27	28	GPIO_88
GPIO_111	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
GPIO_110	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	GPIO_81
AIN6	35	36	AIN5	GPIO_8	35	36	GPIO_80
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	GPIO_7	GPIO_74	41	42	GPIO_75
DGND	43	44	DGND	GPIO_72	43	44	GPIO_73
DGND	45	46	DGND	GPIO_70	45	46	GPIO_71

Tabla 4 – Pines de control ADC

Los argumentos de los ejecutables se corresponden con los siguientes pines de control ADC:

Pines de control ADC	Argumento del ejecutable
AIN0	0
AIN1	1
AIN2	2
AIN3	3
AIN4	4
AIN5	5
AIN6	6

Tabla 5 - Argumentos del ejecutable ADC

4.4.1 leerADC.cpp

La compilación de este archivo con la librería anteriormente mencionada genera el ejecutable leerADC.
Este ejecutable recibe por argumento el pin de control ADC, lee los valores de corriente y hace una conversión para mostrar los valores.

4.4.2 pararADC.cpp

La compilación de este archivo con la librería anteriormente mencionada genera el ejecutable pararADC.
Este ejecutable recibe por argumento el pin de control ADC y desactiva el modo de funcionamiento dejándolo en el modo inicial.

4.5 I2C

En el desarrollo de este modo de funcionamiento conectamos una memoria de la marca ST modelo M24C01 de 1 Kbit serial I²C bus EEPROM a los pines del microcontrolador Beaglebone Black según la siguiente figura.
De esta manera se puede escribir y leer datos en la memoria.

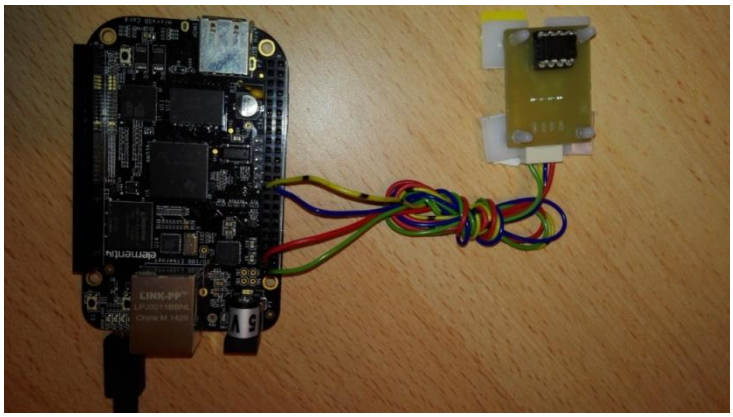


Figura 13 - Desarrollo modo funcionamiento I2C

Para controlar este modo de funcionamiento utilizamos la librería de código y distribución libre BlackLib, la podemos encontrar en la página oficial de beagleboard. [24]

Para saber que pines pueden ser utilizados en este modo I2C, utilizaremos la siguiente tabla.

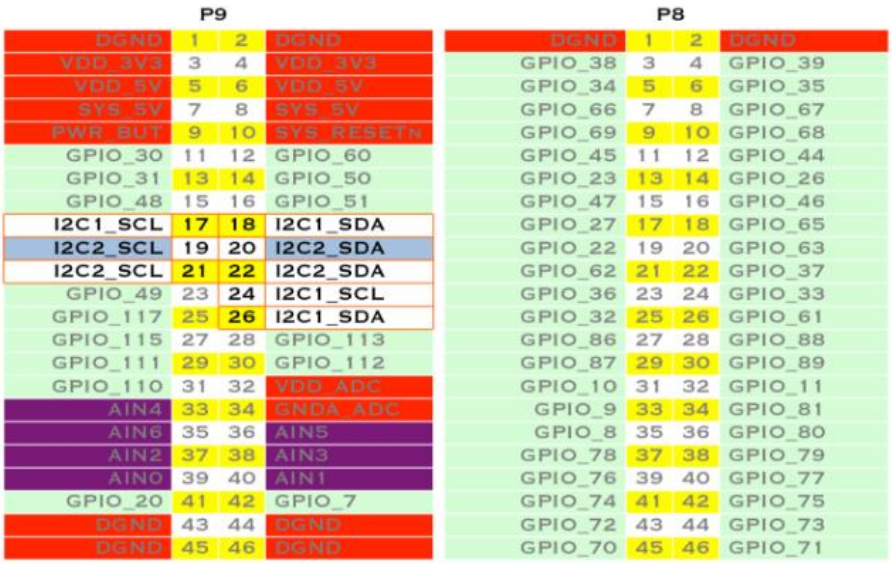


Tabla 6 – Pines de control I2C

Los argumentos de los ejecutables se corresponden con los siguientes pines de control I2C:

Pines de control I2C	Argumento del ejecutable
I2C_1	0
I2C_2	1

Tabla 7 - Argumentos del ejecutable I2C

4.5.1 escribirI2C.cpp

La compilación de este archivo con la librería anteriormente mencionada genera el ejecutable escribirI2C.
Este ejecutable recibe por argumento el pin de control I2C, la dirección en hexadecimal del dispositivo conectado, la dirección en hexadecimal del registro y el valor en hexadecimal que se escribirá.

4.5.2 leerI2C.cpp

La compilación de este archivo con la librería anteriormente mencionada genera el ejecutable escribirI2C.
Este ejecutable recibe por argumento el pin de control I2C, la dirección en hexadecimal del dispositivo conectado y la dirección en hexadecimal del registro que va a ser leído.

4.6 Pantalla LCD

Como aplicación adicional a nuestra plataforma de control, se ha implementado un módulo de pantalla LCD para enviar mensaje desde la web principal a la pantalla LCD 16x2.
.

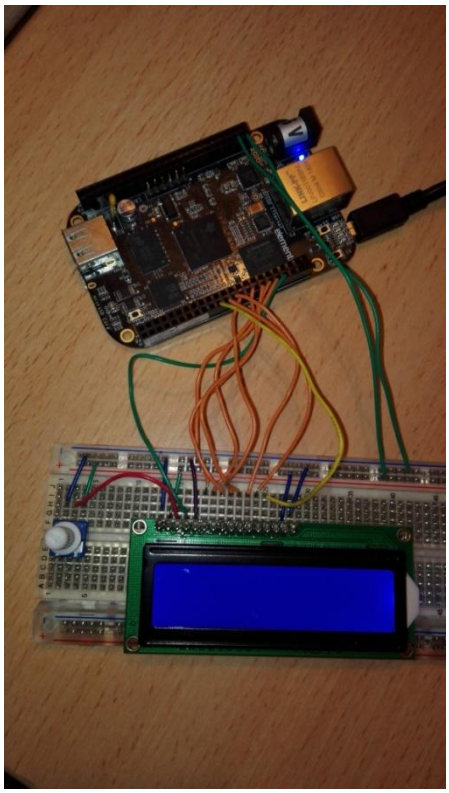


Figura 14 – Desarrollo de pantalla LCD 16x2

Para controlar este modo de funcionamiento he utilizado la librería de código y distribución libre Adafuit, la podemos encontrar en la página oficial de beagleboard. [25]

4.6.1 mensajeLCD.py

Este ejecutable desarrollado en Python, hace uso de la librería anterior, y recibe por argumento el mensaje que será enviado a la pantalla LCD.

5 Integración, pruebas y resultados

En este capítulo se detallan las pruebas realizadas para comprobar todos los modos de funcionamiento, así como la integración en la web.

5.1 Pruebas individuales

Antes de englobar todos los modos de funcionamiento en la web, se han realizado varias pruebas previas para confirmar su funcionamiento.

Estas pruebas consisten en comprobar desde consola, (mediante el programa putty.exe) que los ejecutable realizan la función comentada en el apartado de desarrollo. Es importante habilitar todos los privilegios a los ejecutables con el comando chmod 7777 desde consola.

5.1.1 Pruebas GPIO

Se ha hecho la prueba de encender un LED conectado en el pin P9_11 bajo el siguiente comando en consola.

sudo ./encenderGPIO 30

El LED se enciende.

Se apaga el LED bajo el siguiente comando en consola.

sudo ./apagarGPIO 30

El LED se apaga.

Se hacer la prueba de intermitencia bajo el siguiente comando en consola.

sudo ./intermitenteGPIO 30

El LED parpadea indefinidamente.

Finalmente el LED lo apago bajo el comando apagarGPIO de nuevo. El LED se apaga.

5.1.2 Pruebas PWM

Para comprobar el funcionamiento, además de ejecutar desde consola los ejecutables, también se conectó a un osciloscopio la placa y se generaron las siguientes señales con diferentes periodos y ciclos de trabajo.

Conectamos la sonda del osciloscopio al pin P9_14 y ejecutamos los siguientes comandos:

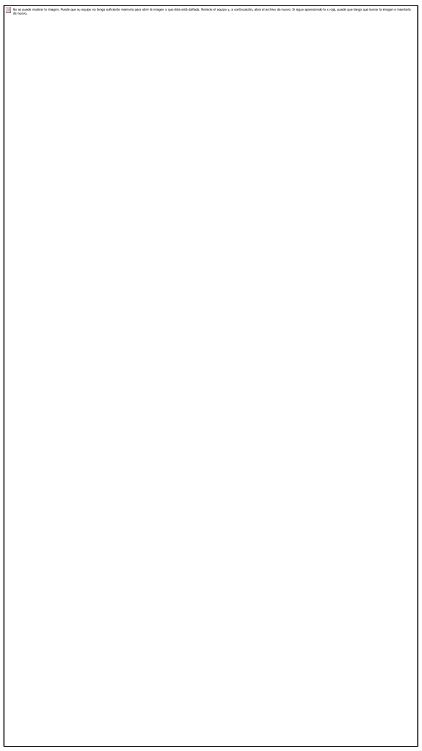


Figura 15 – Prueba conexión PWM

- Periodo 5000 nanosegundos y duty cicle 10% bajo el siguiente comando:

sudo ./activarPWM 2 5000 10

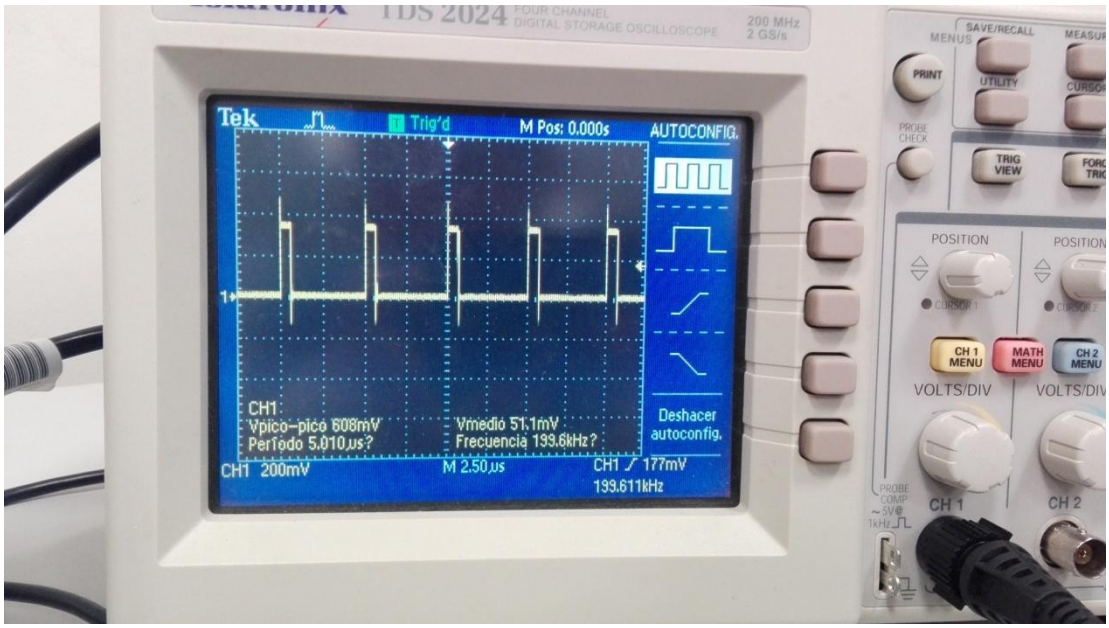


Figura 16 – Prueba osciloscopio PWM 1

- Periodo 5000 nanosegundos y duty cicle 50% bajo el siguiente comando:

sudo ./activarPWM 2 5000 50

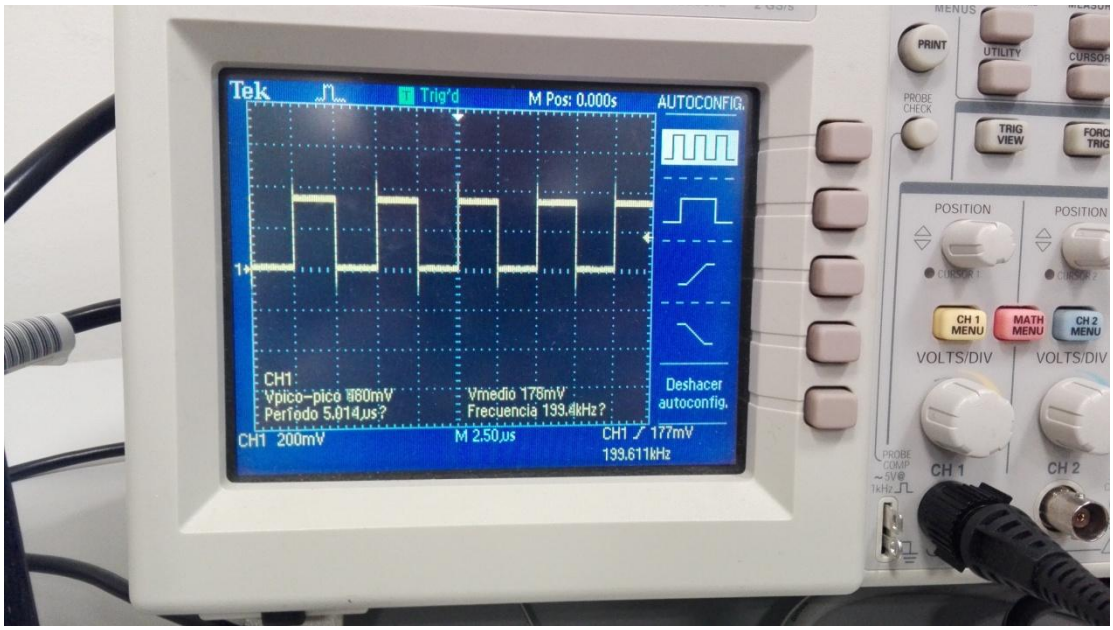


Figura 17 - Prueba osciloscopio PWM 2

- Periodo 1000000 nanosegundos y duty cicle 10% bajo el siguiente comando:

sudo ./activarPWM 2 1000000 10

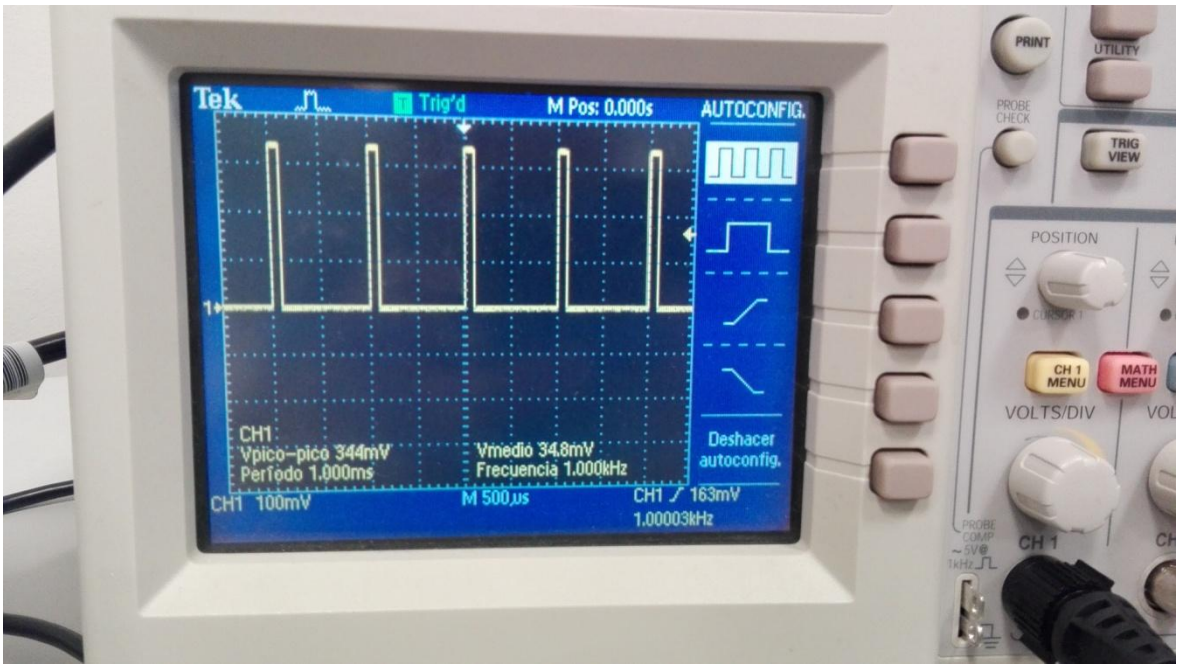


Figura 18 - Prueba osciloscopio PWM 3

- Periodo 1000000 nanosegundos y duty cicle 50% bajo el siguiente comando:

sudo ./activarPWM 2 1000000 50

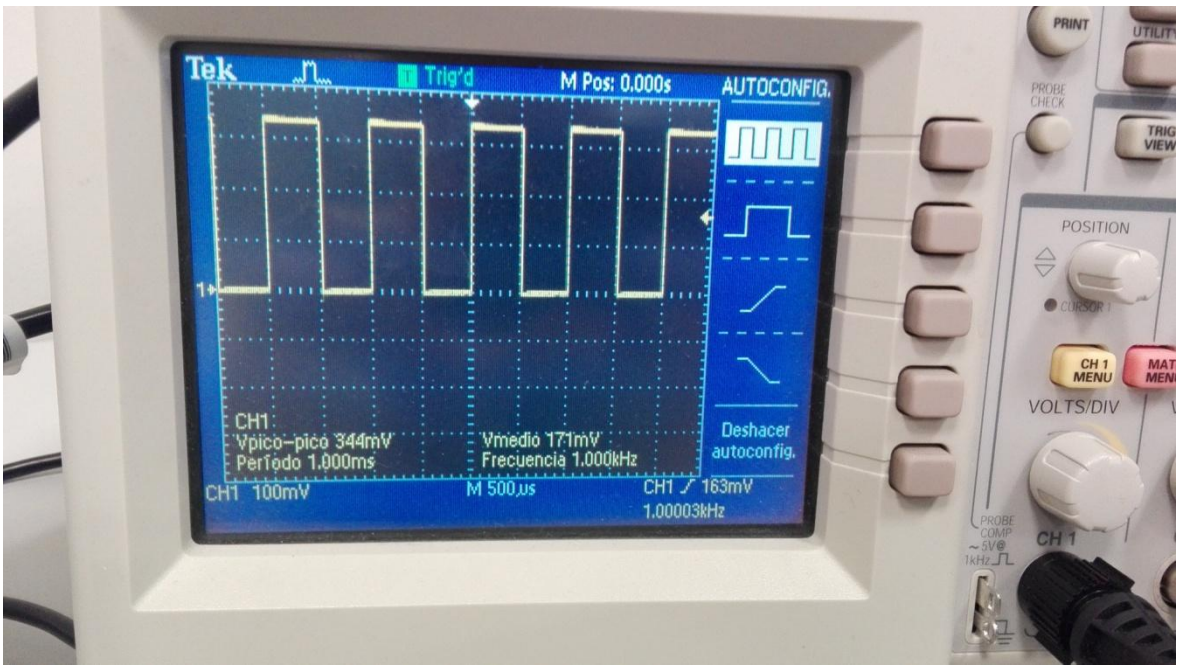


Figura 19 -Prueba osciloscopio PWM 4

5.1.3 Pruebas ADC

Conectamos al pin P9_37un potenciómetro según la figura 12 del capítulo de desarrollo y ejecutamos el siguiente comando para leer la tensión del mismo mientras cambiamos de posición la ruleta:

sudo ./leerADC 2

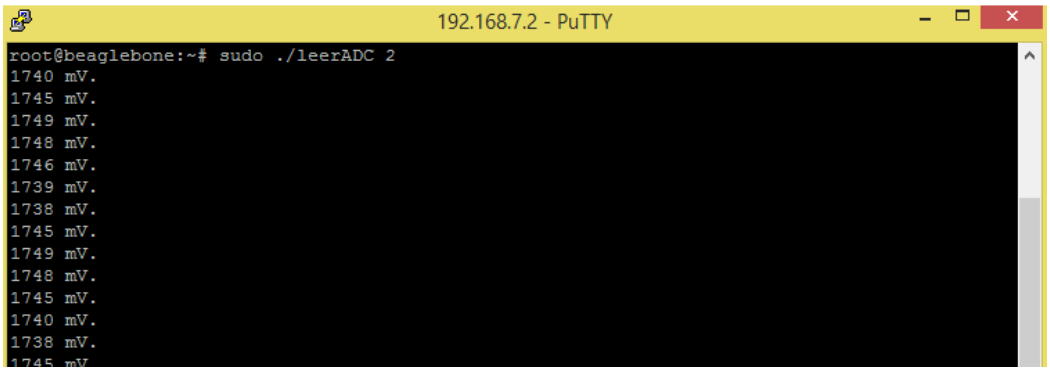


Figura 20 – Pruebas potenciómetro ADC

Se leen los datos mientras cambiamos de posición.

Finalmente paramos de leer datos con el siguiente comando desde consola:

sudo ./pararADC 2

5.1.4 Pruebas I2C

Conectamos la memoria EEPROM de la figura 13 del capítulo de desarrollo a los pines de la siguiente manera, P9_19 a SCL y P9_20 a SDA.

Ejecutamos el siguiente comando bajo consola para escribir en el dispositivo 0x60, en el registro 0x10 el valor 0x20.

sudo ./escribirI2C 0x60 0x10 0x02

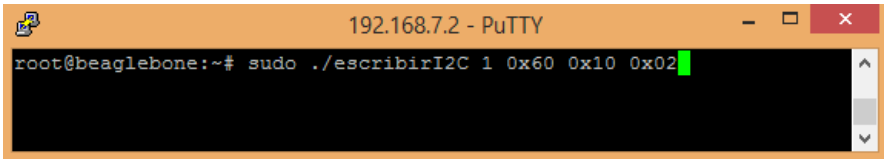


Figura 21 – Pruebas I2C

Procedemos a leer el dato escrito bajo el siguiente comando:

sudo ./leerI2C 0x60 0x10

5.2 Pruebas globales

Estas pruebas consisten en comprobar que desde la web (inicio.php), los ejecutables realizan la función comentada en el apartado de desarrollo.

Para ello, se han ido haciendo llamadas desde la web comprobando los diferentes modos de funcionamiento. De tal forma que si pulsamos sobre el botón asociado al ejecutable, esperemos el resultado esperado de cada modo.

Para el modo de funcionamiento GPIO, se escribe un número de control de pin en el cuadro de texto, se pulsa el botón de encender y el LED se apaga. Se pulsa el botón de apagar y el LED se apaga. Se pulsa el botón de intermitente y el LED parpadea.
Todos estos resultados son correctos.

Para el modo de funcionamiento PWM, se escribe un número de control de pin en el cuadro de texto, se escribe un periodo en el cuadro de texto y un ciclo de trabajo en el cuadro de texto. Se pulsa en el botón activar y el LED conectado al pin se ilumina con cierta luminosidad. Se cambia el valor de ciclo de trabajo a uno más alto y la luminosidad del LED aumenta. Finalmente se pulsa en Parar y el LED se apaga.
Todos estos resultados son correctos.

Para el modo de funcionamiento ADC, se escribe un numero de control de pin en el cuadro de texto y se pulsa el botón leer datos ADC. Se ve como el valor cambia si cambiamos de posición el potenciómetro. Finalmente pulsamos parar ADC.
Todos estos resultados son correctos.

Para el modo de funcionamiento I2C, se escribe la dirección en hexadecimal del dispositivo conectado en el cuadro de texto, se escribe la dirección en hexadecimal del registro en el cuadro de texto y se escribe el valor a escribir en el cuadro de texto. Se pulsa el botón escribir datos.
Posteriormente, se escribe en el cuadro de texto la dirección en hexadecimal del registro y se escribe en el cuadro de texto el registro que queremos leer. Pulsamos sobre el botón leer datos. El dato que fue escrito se puede leer.
Todos estos resultados son correctos.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Basándose en el objetivo de este proyecto, que era el de desarrollar un sistema empotrado para el control de sistemas electrónicos a través de internet, se puede concluir que se ha conseguido desarrollar una serie de ejecutables capaces de controlar sistemas electrónicos conectados nuestro sistema empotrado.

Los ejecutables se han desarrollado usando librerías de código y distribución libre para el manejo de los modos de funcionamiento del sistema embebido Beaglebone Black, del que se han explicado las principales funcionalidades.

Para manejar todo esto de manera remota, se ha explicado y desarrollado un servidor web que ha sido alojado en el sistema empotrado, así como el desarrollo web del código de la página principal.

Con este desarrollo y con ayuda de los manuales que se pueden encontrar en el anexo, se puede construir una página web personalizada para una aplicación en concreto, permitiendo a una persona sin apenas conocimientos de programación, manejar sistemas electrónicos de manera remota.

6.2 Trabajo futuro

Un posible trabajo futuro seria la implementación de los restantes modos de funcionamiento de Beaglebone Black, tales como SPI, UART o PRU.

También sería conveniente elaborar un buen protocolo de seguridad de acceso a la plataforma web. Ya que si hay elementos sensibles conectados a la placa, cualquiera podría manipular de forma inadecuada.

Finalmente se podría adaptar el servidor web para que solo hubiese una conexión, un usuario conectado y evitar bloqueos de usuarios.

Referencias

[1] Domótica.www.cedom.es/sobre-domotica/que-es-domotica
[2] X10. www.xtend.com.ar/queesx10.asp
[3] Hue – Phillips. www2.meethue.com/es-es/
[4] Sorel. www.sorel.de
[5] Derek Molloy, ”Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux”, Ed.Wiley
[6] Perestrelo, R. S.-L. (2015). Beaglebone for dummies.
[7] Sergio Martín Casco, “Raspberry Pi, Arduino y Beaglebone Black Comparación y Aplicaciones” Universidad Católica Nuestra Señora de la Asunción, Facultad de Ciencias y Tecnología, Septiembre 2014, http://jeuazarru.com/wpcontent/ uploads/ 2014/10/MiniPCs.pdf
[8] Richardson, M. (2013). Getting Started with BeagleBone, Linux Powered Electronic Projects with Python and JavaScript.
[9] Raspberry boards, www.raspberrypi.org/
[10] Beaglebone,www.beagleboard.org/
[11] GPIO. Documentación didáctica de la asignatura de Sistemas electrónicos digitales. Universidad Autónoma de Madrid. (2014).
[12] PWM. http://www.ibertronica.es/blog/tutoriales/funcion-pwm/
[13] ADC. http://whatis.techtarget.com/definition/analog-to-digital-conversion-ADC
[14] I2C. https://learn.sparkfun.com/tutorials/i2c
[15] Installing a LAMP server on beagleboard!, http://angrybeagle.blog.com/2011/05/11/ installing-a-lamp-server-on-beagleboard/
[16] Imagen BeagleBone Black. http://beagleboard.org/latest-images
[17] Contenido dinámico con CGI, http://quark.fe.up.pt/apache1.3-es/howto/cgi.html
[18] MySQL. http://searchdatacenter.techtarget.com/es/definicion/MySQL
[19] Run PHP with CGI and Apache on Debian 7 (Wheezy), https://www.linode.com /docs/websites/apache/run-php-cgi-apapache-debian-7
[20] Python. https://www.python.org/
[21] C++. http://c.conclase.net/
[22] Ejecución de programas del sistema, http://php.net/manual/es/book.exec.php.
[23] Beaglebone Web Server, http://kingofprotons.blogspot.com.es/p/beaglebone-web-server.html
[24] Librería BlackLib. http://blacklib.yigityuce.com/
[25] Librería Adafuit. https://www.adafruit.com/
[26] Contenido didáctico del curso Sistemas Embebidos. Universidad Nacional Abierta y a Distancia de Colombia.
[27] Beagleboard. BeagleBone Rev A6 System Reference Manual. (May, 2012).
[28] Chapter 8 - The Debian package management tools, https://www.debian.org/doc/ manuals/debian-faq/ch-pkgtools.en.html
[29] Beaglebone Black, Distribución Linux. (http://elinux.org/BeagleBone).

Glosario

EPS	Escuela Politécnica Superior
UAM	Universidad Autónoma de Madrid
BBB	BeagleBone Black
SO	Sistema Operativo
OCP	On Chip Peripherals
GPIO	General Purpose Input Output
LED	Light-Emitting Diode
PWM	Pulse Width Modulation
ADC	Analog-to-Digital Conversion
I2C	Inter-Integrated Circuit
SDA	Serial Data
SCL	Serial Clock

Anexos

A Manual de instalación

Inicialmente, la BeagleBone Black viene configurada de fábrica con una serie elementos que deben ser cambiados y actualizados para conseguir el objetivo de este proyecto.

Instalación de Drivers

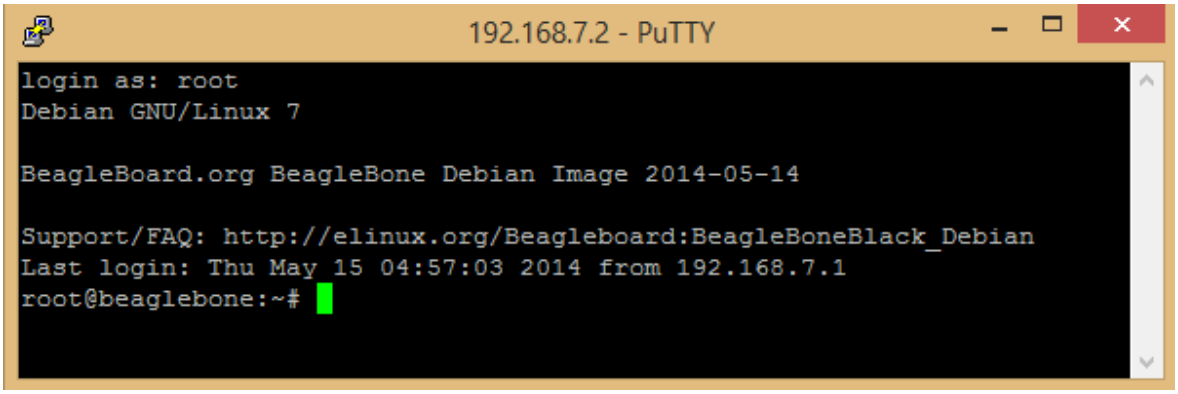
El primer paso es el de conectar la BBB con el cable USB a un PC. Tras leerse la tarjeta de memoria en el ordenador, tendremos acceso a diferentes accesos directos del microcontrolador como ficheros de primera lectura, gettingstarts, readmes... Pulsaremos en el icono de Beagleboard para instalar los drivers necesarios. Instalaremos según nuestro sistema operativo el controlador adecuado.

Conexión a través del protocoloSSH

El siguiente paso será el de conectarnos de forma remota a la BBB a través del protocolo SSH. Esto es posible, gracias a que la BBB crea virtualmente una red a través del cable USB.
La conexión a través de este protocolo puede hacerse de varias maneras. Se ha optado por hacer la conexión a través del programa PUTTY.exe.

Tras arrancar este programa, introducir el “host name” o la dirección IP a la que queremos conectar, el puerto correspondiente y seleccionar conectar.
En el caso de la BBB, la dirección IP por defecto es 192.168.7.2 y el puerto 22.

Tras conectar, aparecerá la terminal del programa, entraremos como usuario “root” y sin contraseña vacía. Aparecerá la siguiente información.



Tras acceder como usuario “root” se puede ver en la terminal el sistema operativo, la ultima fecha de login y la imagen instalada.

Conexión a internet

El siguiente paso es el de actualizar el sistema operativo instalado, para ello necesitaremos conectar a internet la BBB.
Se muestran a continuación las siguientes opciones:

1. Podremos conectarnos a Internet a través de un cable de red conectado a través del conector RJ-45 configurando la dirección IP del cable que conectamos al puerto.

Para ello, introducir los siguientes comandos:

Sudo nano etc/networks/interfaces

En el fichero de configuración introducir el siguiente código.

```
iface eth0 inet static
      address 192.168.7.2
netmask 255.255.255.0
      gateway 192.168.1.254
dns-nameservers 8.8.8.8
dns-nameservers 8.8.4.4
```

Comentar esta parte del código, que es la encargada de generar una red a través del cable USB.

```
#iface usb0 inet static
#    address 192.168.7.2
#    netmask 255.255.255.0
#    network 192.168.7.0
#    gateway 192.168.7.1
```

Guardar, salir y reiniciar la BBB.

2. Otra forma de conectarse a internet es a través de un adaptador de red conectado a la entrada USB de la BBB.

Para ello, seguir los siguientes pasos:

- Conectar el adaptador USB en la entrada USB.
- Conectar la alimentación de 5V de la BBB.
- Introducir los siguientes comandos:

sudo lsusb– Verificar que el dispositivo ha sido reconocido por la BBB.

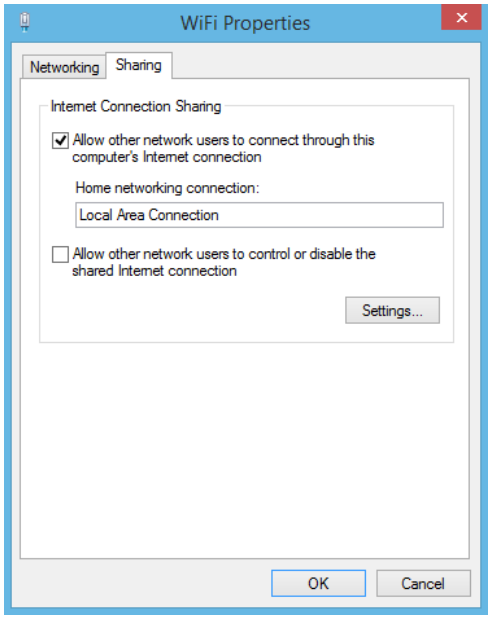
sudo iwconfig – Comprobar parámetros de la configuración inalámbrica.

sudo wicdcourses – Seleccionar la red a conectar e introducir la contraseña.

3. Compartir la conexión wifi del PC con la BBB a través del cable USB.

En este proyecto se ha optado por esta última opción, y el procedimiento es el siguiente:

- Desde el panel de control del PC, en centro de redes y recursos compartidos, seleccionar la conexión wifi del PC y seleccionar las propiedades.
- En la interface, seleccionar la pestacha de compartir y elegir la opción de Local AreaConnection.
- Seleccionar "OK".



Actualización y mejora de la BBB

Una vez que ya se tiene conexión a internet, el siguiente paso es el de actualizar y mejorar la BBB.
Para ello, escribir el siguiente código desde la terminal:

```
cd ..
apt-get update
apt-get upgrade
```

El proceso tardará varios minutos hasta que quede totalmente actualizada y mejorada.
Si la conexión a internet fallase, se tendría que volver a realizar los últimos 2 pasos.

Configuración de parámetros

BBB tiene internamente instalado un interfaz de desarrollo, Cloud9 IDE, este interfaz nos permite escribir código escribiendo la dirección 192.168.7.2:8080 desde un navegador, compilar el código y ejecutarlo.

Nuestro objetivo es el de crear el código, compilarlo y ejecutarlo independientemente del entorno de desarrollo que usemos. En nuestro caso desarrollaremos código en C++ mediante consola, compilaremos directamente sobre la placa y ejecutaremos desde un navegador web el ejecutable generado.

Para que esto funcione y el entorno de desarrollo no interfiera en nuestro código, tenemos que desactivarlo, además de otros parámetros y servicios para garantizar que el puerto 80 está libre para servir nuestro nuevo sitio web.

Desde la terminal, escribiremos:

```
systemctl disable bonescript.service
systemctl disable bonescript.socket
systemctl disable bonescript-autorun.service
systemctl disable avahi-daemon.service
systemctl disable cloud9.service
systemctl disable gateone.service
systemctl disable gdm.service
systemctl disable mpd.service
```

Una vez deshabilitados estos servicios, ya podremos empezar a construir un servidor web que nos permita ejecutar el código compiladoen el navegador.

Instalación del servidor web

Para instalar **Apache** en la BBB, seguir los siguientes pasos:

```
sudo apt-getinstall apache2
```

Para comprobar si se ha instalado Apache correctamente, escribiremos en el navegador la dirección IP del servidor, en este caso el de la BBB (http://192.168.7.2:8080). La página debe mostrar el mensaje "Itworks!".

Dirección IP de la BBB.

Para saber la dirección IP del servidor, en este caso el de la BBB, ejecutar el siguiente comando en la terminal.

```
ifconfig eth0 | inet grep | awk '{print $ 2}'
```

Instalación MySQL

Para instalar MySQL, escribir los siguientes comandos en la consola:

sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql

Durante la instalación, MySQLpedirá establecer una contraseña. Si se pasara la oportunidad de establecer la contraseña mientras el programa se está instalando, es muy fácil de configurar la contraseña más tarde desde dentro de la interface de MySQL.

Una vez que se ha instalado MySQL, debemos activarlo con este comando:

sudo mysql_install_db

Finalmente, escribir esto:

sudo / usr / bin / mysql_secure_installation

El siguiente mensaje le pedirá la contraseña de root actual. Si no se ha cambiado la contraseña, por defecto está vacía.

Seleccionar “Yes” en los siguientes y finalizar la instalación.

Instalación PHP

Para instalar PHP, escribir en la terminal.

sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt

Después de responder afirmativamente al mensaje dos veces, PHP se instalará.

Una vez instalado, vamos al crear el índice del directorio, para servir los archivos de índice de php relevantes.

Para ello, escribir en la terminal:

/etc/apache2/mods-enabled/dir.conf sudo nano

Añadir index.php al principio del archivo de índices. La página debería tener el siguiente aspecto:

<IfModulemod_dir.c>

DirectoryIndexindex.php index.html index.phpindex.cgi index.pl index.xhtml index.htm

</ IfModule>

El siguiente paso será el de instalar módulos de extensión para completar el servidor web.

Para ello, escribir en consola el siguiente comando:

sudo apt-getinstall php5-cgi php5-cli php5-dbg php5-dev php5-mysql

Estos módulos, servirán para manejar lenguaje de script embebido (php5-cgi), interpretar la línea de comandos para el lenguaje de scripting (php5-cli), archivos para debugear (php5-dbg) o archivos para desarrollar (php5-dev).

Finalmente, para saber si está correctamente instalado PHP realizar los siguientes pasos:

Escribir en consola el siguiente comando:

/var/www/info.php sudo nano

En la interface de la edición del código, añadir las siguientes líneas:

<? php

phpinfo ();

?>

A continuación, guardar y salir.

Reiniciar Apache para que todos los cambios surtan efecto:

sudo service apache2 restart

Terminar visitando la página de información PHP (http://192.168.7.2:8080/info.php)

Si todo va bien, tendrá un aspecto como en la siguiente figura.

PHP Version 5.4.45-0+deb7u3

System	Linux beaglebone 3.8.13-bone50 #1 SMP Tue May 13 13:24:52 UTC 2014 armv7l
Build Date	May 31 2016 18:06:02
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-curl.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-imap.ini, /etc/php5/apache2/conf.d/20-intl.ini, /etc/php5/apache2/conf.d/20-mcrypt.ini, /etc/php5/apache2/conf.d/20-memcache.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-pdo_sqlite.ini, /etc/php5/apache2/conf.d/20-pspell.ini, /etc/php5/apache2/conf.d/20-recode.ini, /etc/php5/apache2/conf.d/20-snmp.ini, /etc/php5/apache2/conf.d/20-sqlite3.ini, /etc/php5/apache2/conf.d/20-tidy.ini, /etc/php5/apache2/conf.d/20-xmlrpc.ini, /etc/php5/apache2/conf.d/imagick.ini, /etc/php5/apache2/conf.d/ming.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525

B Manual del programador

En este documento se explica en detalle cómo se debe configurar la plataforma para una aplicación en concreto.
En primer lugar, destacar que solo será posible construir aplicaciones que necesiten los siguientes modos de funcionamiento:

- GPIO
- PWM
- ADC
- I2C

De esta manera, y sabiendo los modos que necesitaremos en la aplicación, editaremos el archivo inicio.php. Para acceder a estos archivos, será necesario conectarse mediando SSH a través del programa WinSCP y editarlo con un editor de texto. (Consultar manual de instalación)

En el archivo inicio.php están incluidas todas las llamadas a todos los ejecutables de los modos de funcionamiento disponibles.

Estos son los ejemplo a los ejecutables que contiene el archivo inicio.php:

Modo GPIO

- **encenderGPIO.cpp**

Este ejecutable se ejecuta de la siguiente manera: sudo ./encenderGPIO "numero de pin de control", siendo el número de pin el indicado en las tablas del capítulo de desarrollo.

Por ejemplo, si queremos activar el pin P9_12 introduciremos el valor 60.

sudo ./encenderGPIO 60

- **apagarGPIO.cpp**

Este ejecutable se ejecuta de la siguiente manera: sudo ./apagarGPIO "numero de pin de control", siendo el número de pin el indicado en las tablas del capítulo de desarrollo.

Por ejemplo, si queremos desactivar el pin P9_12 introduciremos el valor 60.

sudo ./apagarGPIO 60

- **intermitenteGPIO.cpp**

Este ejecutable se ejecuta de la siguiente manera: sudo ./intermitenteGPIO "numero de pin de control", siendo el número de pin el indicado en las tablas del capítulo de desarrollo.

Por ejemplo, si queremos activar y desactivar el pin P9_12 introduciremos el valor 60.

sudo ./intermitenteGPIO 60

Modo PWM

- **activarPWM.cpp**

Este ejecutable se ejecuta de la siguiente manera: sudo ./activarPWM "numero de pin de control" "periodo" "ciclo de trabajo", siendo el número de pin el indicado en las tablas del capítulo de desarrollo.

Por ejemplo, si queremos generar una señal pwm en el pin P9_14, introduciremos el valor 2, periodo de 5000 nanosegundos y ciclo de trabajo al 50%.

sudo ./activarPWM 2 5000 50

- **pararPWM.cpp**

Este ejecutable se ejecuta de la siguiente manera: sudo ./pararPWM "numero de pin de control" ,siendo el número de pin el indicado en las tablas del capítulo de desarrollo.

Por ejemplo, si queremos parar la señal pwm en el pin P9_14, introduciremos el valor 2.

sudo ./pararPWM 2

Modo ADC

- **leerADC.cpp**

Este ejecutable se ejecuta de la siguiente manera: sudo ./leerADC "numero de pin de control" ,siendo el número de pin el indicado en las tablas del capítulo de desarrollo.

Por ejemplo, si queremos leer los datos del pin P9_37, introduciremos el valor 2 .

sudo ./leerADC 2

- **pararADC.cpp**

Este ejecutable se ejecuta de la siguiente manera: sudo ./paraADC "numero de pin de control" ,siendo el número de pin el indicado en las tablas del capítulo de desarrollo.

Por ejemplo, si queremos dejar de leer los datos del pin P9_37, introduciremos el valor 2 .

sudo ./pararADC 2

Modo I2C

- **escribirI2C.cpp**

Este ejecutable se ejecuta de la siguiente manera: sudo ./escribirI2C “numero de pin de control” “dirección en hexadecimal del dispositivo conectado” “dirección en hexadecimal del registro” “valor en hexadecimal que se escribirá” ,siendo el número de pin el indicado en las tablas del capítulo de desarrollo.

Por ejemplo, si queremos escribir datos a través del bus I2C_1, introduciremos el valor 1, a la dirección del dispositivo conectado 0x60, a la dirección del registro 0x10 y con valor 0x02.

sudo ./escribirI2C 1 0x60 0x10 0x02

- **leerI2C.cpp**

Este ejecutable se ejecuta de la siguiente manera: sudo ./leerI2C “numero de pin de control” “dirección en hexadecimal del dispositivo conectado” “dirección en hexadecimal del registro”,siendo el número de pin el indicado en las tablas del capítulo de desarrollo.

Por ejemplo, si queremos leer datos a través del bus I2C_1, introduciremos el valor 1, a la dirección del dispositivo conectado 0x60 y a la dirección del registro 0x10.

sudo ./leerI2C 1 0x60 0x10

Pantalla LCD

- **mensajeLCD.py**

Este ejecutable se ejecuta de la siguiente manera: sudo pyhon mensajeLCD.py “texto a enviar”

Por ejemplo, si queremos escribir el mensaje “hola” en la pantalla LCD, escribiremos.

sudo pyhon mensajeLCD.py hola

C Cabeceras de pines P8 y P9

P9 Pin	Offset	mode0	mode1	mode2	mode3	mode4	mode5	mode6	mode7	GPIO #	P9 Pin
P9_1	GND										P9_1
P9_2	GND										P9_2
P9_3	3.3V										P9_3
P9_4	3.3V										P9_4
P9_5	VDD_5V										P9_5
P9_6	VDD_5V										P9_6
P9_7	SYS_5V										P9_7
P9_8	SYS_5V										P9_8
P9_9	PWR_BUT										P9_9
P9_10		Reset_Out									P9_10
P9_11	0x870	gpmc_wait0	gmii2_crs	gpmc_csn4	rmii2_crs_dv	mmc1_sdcd	pr1_mii1_col	uart4_rxd	gpio0_30	30	P9_11
P9_12	0x878	gpmc_be1n	gmii2_col	gpmc_csn6	mmc2_dat3	gpmc_dir	pr1_mii1_rdlink	mcasp0_aclkr	gpio1_28	60	P9_12
P9_13	0x874	gpmc_wpn	gmii2_rxer	gpmc_csn5	rmii2_rxer	mmc2_sdcd	pr1_mii1_txen	uart4_txd	gpio0_31	31	P9_13
P9_14	0x848	gpmc_a2	gmii2_txd3	rgmii2_txd3	mmc2_dat1	gpmc_a18	pr1_mii1_txd2	ehrpwm1A	gpio1_18	50	P9_14
P9_15	0x840	gpmc_a0	gmii2_txen	rgmii2_tctl	rmii2_txen	gpmc_a16	pr1_mii_mt1_clk	ehrpwm1_tripzone_input	gpio1_16	48	P9_15
P9_16	0x84C	gpmc_a3	gmii2_txd2	rgmii2_txd2	mmc2_dat2	gpmc_a19	pr1_mii1_txd1	ehrpwm1B	gpio1_19	51	P9_16
P9_17	0x95C	spi0_cs0	mmc2_sdwp	I2C1_SCL	ehrpwm0_synci	pr1_uart0_txd	pr1_edio_data_in1	pr1_edio_data_out1	gpio0_5	5	P9_17
P9_18	0x958	spi0_d1	mmc1_sdwp	I2C1_SDA	ehrpwm0_tripzone_input	pr1_uart0_rxd	pr1_edio_data_in0	pr1_edio_data_out0	gpio0_4	4	P9_18
P9_19	0x97C	uart1_rtsn	timer5	dcan0_rx	I2C2_SCL	spi1_cs1	pr1_uart0_rts_n	pr1_edc_latch1_in	gpio0_13	13	P9_19
P9_20	0x978	uart1_ctsn	timer6	dcan0_tx	I2C2_SDA	spi1_cs0	pr1_uart0_cts_n	pr1_edc_latch0_in	gpio0_12	12	P9_20
P9_21	0x954	spi0_d0	uart2_txd	I2C2_SCL	ehrpwm0B	pr1_uart0_rts_n	pr1_edio_latch_in	EMU3	gpio0_3	3	P9_21
P9_22	0x950	spi0_sclk	uart2_rxd	I2C2_SDA	ehrpwm0A	pr1_uart0_cts_n	pr1_edio_sof	EMU2	gpio0_2	2	P9_22
P9_23	0x844	gpmc_a1	gmii2_rxdv	rgmii2_rctl	mmc2_dat0	gpmc_a17	pr1_mii1_txd3	ehrpwm0_synco	gpio1_17	49	P9_23
P9_24	0x984	uart1_txd	mmc2_sdwp	dcan1_rx	I2C1_SCL	EMU4	pr1_uart0_txd	pr1_pru0_pru_r31_16	gpio0_15	15	P9_24
P9_25	0x9AC	mcasp0_ahclkx	eQEP0_strobe	mcasp0_axr3	mcasp1_axr1	EMU4	pr1_pru0_pru_r30_7	pr1_pru0_pru_r31_7	gpio3_21	117	P9_25
P9_26	0x980	uart1_rxd	mmc1_sdwp	dcan1_tx	I2C1_SDA		pr1_uart0_rxd	pr1_pru1_pru_r31_16	gpio0_14	14	P9_26
P9_27	0x9A4	mcasp0_fsr	eQEP0B_in	mcasp0_axr3	mcasp1_fsx	EMU2	pr1_pru0_pru_r30_5	pr1_pru0_pru_r31_5	gpio3_19	115	P9_27
P9_28	0x99C	mcasp0_ahclkr	ehrpwm0_synci	mcasp0_axr2	spi1_cs0	eCAP2_in_PWM2_out	pr1_pru0_pru_r30_3	pr1_pru0_pru_r31_3	gpio3_17	113	P9_28
P9_29	0x994	mcasp0_fsx	ehrpwm0B		spi1_d0	mmc1_sdcd	pr1_pru0_pru_r30_1	pr1_pru0_pru_r31_1	gpio3_15	111	P9_29
P9_30	0x998	mcasp0_axr0	ehrpwm0_tripzone_input		spi1_d1	mmc2_sdcd	pr1_pru0_pru_r30_2	pr1_pru0_pru_r31_2	gpio3_16	112	P9_30
P9_31	0x990	mcasp0_aclkx	ehrpwm0A		spi1_sclk	mmc0_sdcd	pr1_pru0_pru_r30_0	pr1_pru0_pru_r31_0	gpio3_14	110	P9_31
P9_32	VADC										P9_32
P9_33		AIN4									P9_33
P9_34	AGND										P9_34
P9_35		AIN6									P9_35
P9_36		AIN5									P9_36
P9_37		AIN2									P9_37
P9_38		AIN3									P9_38
P9_39		AIN0									P9_39
P9_40		AIN1									P9_40
P9_41	0x9B4	xdma_event_intr1		tclkin	clkout2	timer7	pr1_pru0_pru_r31_16	EMU3	gpio0_20	20	P9_41
P9_41.1	0x9A8	mcasp0_axr1	eQEP0_index		mcasp1_axr0	EMU3	pr1_pru0_pru_r30_6	pr1_pru0_pru_r31_6	gpio3_20	116	P9_41.1
P9_42	0x964	eCAP0_in_PWM0_out	uart3_txd	spi1_cs1	pr1_ecap0_ecap_capin_apwm_o	spi1_sclk	mmc0_sdwp	xdma_event_intr2	gpio0_7	7	P9_42
P9_42.1	0x9A0	mcasp0_aclkr	eQEP0A_in	mcasp0_axr2	mcasp1_aclkx	mmc0_sdwp	pr1_pru0_pru_r30_4	pr1_pru0_pru_r31_4	gpio3_18	114	P9_42.1
P9_43	GND										P9_43
P9_44	GND										P9_44
P9_45	GND										P9_45
P9_46	GND										P9_46

P8 Pin	Offset	mode0	mode1	mode2	mode3	mode4	mode5	mode6	mode7	GPIO #	P8 Pin.
P8_1	GND										P8_1
P8_2	GND										P8_2
P8_3	0x818	gpmc_ad6	mmc1_dat6						gpio1_6	38	P8_3
P8_4	0x81C	gpmc_ad7	mmc1_dat7						gpio1_7	39	P8_4
P8_5	0x808	gpmc_ad2	mmc1_dat2						gpio1_2	34	P8_5
P8_6	0x80C	gpmc_ad3	mmc1_dat3						gpio1_3	35	P8_6
P8_7	0x890	gpmc_advn_ale		timer4					gpio2_2	66	P8_7
P8_8	0x894	gpmc_oen_ren		timer7					gpio2_3	67	P8_8
P8_9	0x89C	gpmc_be0n_cle		timer5					gpio2_5	69	P8_9
P8_10	0x898	gpmc_wen		timer6					gpio2_4	68	P8_10
P8_11	0x834	gpmc_ad13	lcd_data18	mmc1_dat5	mmc2_dat1	eQEP2B_in	pr1_mii0_txd1	pr1_pru0_pru_r30_15	gpio1_13	45	P8_11
P8_12	0x830	gpmc_ad12	lcd_data19	mmc1_dat4	mmc2_dat0	eQEP2A_in	pr1_mii0_txd2	pr1_pru0_pru_r30_14	gpio1_12	44	P8_12
P8_13	0x824	gpmc_ad9	lcd_data22	mmc1_dat1	mmc2_dat5	ehrpwm2B	pr1_mii0_col		gpio0_23	23	P8_13
P8_14	0x828	gpmc_ad10	lcd_data21	mmc1_dat2	mmc2_dat6	ehrpwm2_tripzone_input	pr1_mii0_txen		gpio0_26	26	P8_14
P8_15	0x83C	gpmc_ad15	lcd_data16	mmc1_dat7	mmc2_dat3	eQEP2_strobe	pr1_ecap0_ecap_capin_apwm_o	pr1_pru0_pru_r31_15	gpio1_15	47	P8_15
P8_16	0x838	gpmc_ad14	lcd_data17	mmc1_dat6	mmc2_dat2	eQEP2_index	pr1_mii0_txd0	pr1_pru0_pru_r31_14	gpio1_14	46	P8_16
P8_17	0x82C	gpmc_ad11	lcd_data20	mmc1_dat3	mmc2_dat7	ehrpwm0_synco	pr1_mii0_txd3		gpio0_27	27	P8_17
P8_18	0x88C	gpmc_clk	lcd_memory_clk	gpmc_wait1	mmc2_clk	pr1_mii1_crs	pr1_mdio_mdclk	mcasp0_fsr	gpio2_1	65	P8_18
P8_19	0x820	gpmc_ad8	lcd_data23	mmc1_dat0	mmc2_dat4	ehrpwm2A	pr1_mii_mt0_clk		gpio0_22	22	P8_19
P8_20	0x884	gpmc_csn2	gpmc_be1n	mmc1_cmd	pr1_edio_data_in7	pr1_edio_data_out7	pr1_pru1_pru_r30_13	pr1_pru1_pru_r31_13	gpio1_31	63	P8_20
P8_21	0x880	gpmc_csn1	gpmc_clk	mmc1_clk	pr1_edio_data_in6	pr1_edio_data_out6	pr1_pru1_pru_r30_12	pr1_pru1_pru_r31_12	gpio1_30	62	P8_21
P8_22	0x814	gpmc_ad5	mmc1_dat5						gpio1_5	37	P8_22
P8_23	0x810	gpmc_ad4	mmc1_dat4						gpio1_4	36	P8_23
P8_24	0x804	gpmc_ad1	mmc1_dat1						gpio1_1	33	P8_24
P8_25	0x800	gpmc_ad0	mmc1_dat0						gpio1_0	32	P8_25
P8_26	0x87C	gpmc_csn0							gpio1_29	61	P8_26
P8_27	0x8E0	lcd_vsync	gpmc_a8	gpmc_a1	pr1_edio_data_in2	pr1_edio_data_out2	pr1_pru1_pru_r30_8	pr1_pru1_pru_r31_8	gpio2_22	86	P8_27
P8_28	0x8E8	lcd_pclk	gpmc_a10	pr1_mii0_crs	pr1_edio_data_in4	pr1_edio_data_out4	pr1_pru1_pru_r30_10	pr1_pru1_pru_r31_10	gpio2_24	88	P8_28
P8_29	0x8E4	lcd_hsync	gpmc_a9	gpmc_a2	pr1_edio_data_in3	pr1_edio_data_out3	pr1_pru1_pru_r30_9	pr1_pru1_pru_r31_9	gpio2_23	87	P8_29
P8_30	0x8EC	lcd_ac_bias_en	gpmc_a11	pr1_mii1_crs	pr1_edio_data_in5	pr1_edio_data_out5	pr1_pru1_pru_r30_11	pr1_pru1_pru_r31_11	gpio2_25	89	P8_30
P8_31	0x8D8	lcd_data14	gpmc_a18	eQEP1_index	mcasp0_axr1	uart5_rxd	pr1_mii_mr0_clk	uart5_ctsn	gpio0_10	10	P8_31
P8_32	0x8DC	lcd_data15	gpmc_a19	eQEP1_strobe	mcasp0_ahclkx	mcasp0_axr3	pr1_mii0_rxdv	uart5_rtsn	gpio0_11	11	P8_32
P8_33	0x8D4	lcd_data13	gpmc_a17	eQEP1B_in	mcasp0_fsr	mcasp0_axr3	pr1_mii0_rxer	uart4_rtsn	gpio0_9	9	P8_33
P8_34	0x8CC	lcd_data11	gpmc_a15	ehrpwm1B	mcasp0_ahclkr	mcasp0_axr2	pr1_mii0_rxd0	uart3_rtsn	gpio2_17	81	P8_34
P8_35	0x8D0	lcd_data12	gpmc_a16	eQEP1A_in	mcasp0_aclkr	mcasp0_axr2	pr1_mii0_rxlink	uart4_ctsn	gpio0_8	8	P8_35
P8_36	0x8C8	lcd_data10	gpmc_a14	ehrpwm1A	mcasp0_axr0		pr1_mii0_rxd1	uart3_ctsn	gpio2_16	80	P8_36
P8_37	0x8C0	lcd_data8	gpmc_a12	ehrpwm1_tripzone_input	mcasp0_aclkx	uart5_txd	pr1_mii0_rxd3	uart2_ctsn	gpio2_14	78	P8_37
P8_38	0x8C4	lcd_data9	gpmc_a13	ehrpwm0_synco	mcasp0_fsx	uart5_rxd	pr1_mii0_rxd2	uart2_rtsn	gpio2_15	79	P8_38
P8_39	0x8B8	lcd_data6	gpmc_a6	pr1_edio_data_in6	eQEP2_index	pr1_edio_data_out6	pr1_pru1_pru_r30_6	pr1_pru1_pru_r31_6	gpio2_12	76	P8_39
P8_40	0x8BC	lcd_data7	gpmc_a7	pr1_edio_data_in7	eQEP2_strobe	pr1_edio_data_out7	pr1_pru1_pru_r30_7	pr1_pru1_pru_r31_7	gpio2_13	77	P8_40
P8_41	0x8B0	lcd_data4	gpmc_a4	pr1_mii0_txd1	eQEP2A_in		pr1_pru1_pru_r30_4	pr1_pru1_pru_r31_4	gpio2_10	74	P8_41
P8_42	0x8B4	lcd_data5	gpmc_a5	pr1_mii0_txd0	eQEP2B_in		pr1_pru1_pru_r30_5	pr1_pru1_pru_r31_5	gpio2_11	75	P8_42
P8_43	0x8A8	lcd_data2	gpmc_a2	pr1_mii0_txd3	ehrpwm2_tripzone_input		pr1_pru1_pru_r30_2	pr1_pru1_pru_r31_2	gpio2_8	72	P8_43
P8_44	0x8AC	lcd_data3	gpmc_a3	pr1_mii0_txd2	ehrpwm0_synco		pr1_pru1_pru_r30_3	pr1_pru1_pru_r31_3	gpio2_9	73	P8_44
P8_45	0x8A0	lcd_data0	gpmc_a0	pr1_mii_mt0_clk	ehrpwm2A		pr1_pru1_pru_r30_0	pr1_pru1_pru_r31_0	gpio2_6	70	P8_45
P8_46	0x8A4	lcd_data1	gpmc_a1	pr1_mii0_txen	ehrpwm2B		pr1_pru1_pru_r30_1	pr1_pru1_pru_r31_1	gpio2_7	71	P8_46
P8 Pin	Offset	mode0	mode1	mode2	mode3	mode4	mode5	mode6	mode7	GPIO #	P8 Pin